UNIVERSITI PUTRA MALAYSIA

ENHANCING STATE-SENSITIVITY PARTITIONING TECHNIQUE BY USING PRIORITIZATION APPROACH

NOR SAADAH BINTI SULAIMAN

FSKTM 2015 36

ENHANCING STATE-SENSITIVITY PARTITIONING TECHNIQUE BY USING PRIORITIZATION APPROACH

By

NOR SAADAH BINTI SULAIMAN

Thesis Submitted To University Putra Malaysia In Fulfilment For The Degree Of Master Computer Science

July 2015

MENINGKATKAN TENIK *STATE-SENSITIVITY PARTITIONING* MENGGUNAKAN PENDEKATAN KEUTAMAAN

Oleh

NOR SAADAH BINTI SULAIMAN

Tesis yang dikemukakan kepada Universiti Putra Malaysia untuk memperolehi Ijazah Sarjana Sains Komputer

Julai 2015

Abstract of thesis presented to University Putra Malaysia in fulfilment of the requirement for the degree of Master Computer Science

ENHANCING STATE-SENSITIVITY PARTITIONING TECHNIQUE USING PRIORITIZATION APPROACH

By

Nor Saadah binti Sulaiman

July 2015

Software Testing is one of the activities in Software Quality Assurance to indicate the successfulness of a project. In dynamic software testing, grey-box testing has been suggested to consider both white-box (structure-based testing) and black-box specification-based testing). State-Sensitivity-Partitioning (SSP) is a new technique to generate test cases for grey-box testing. The lengthy sequence with redundant data state makes the testing process expensive and may lead to inefficiency. Thus, this research proposed a prioritization approach in specifying and reordering the generated SSP test cases to enhance the SSP technique. First, coverage-based information which are code coverage, branch coverage, function coverage and path coverage were collected. We apply a weighted method to calculate the priority value for each test sequence based on the collected coverage information. Test cases then are ranked based on the calculated priority values and thus, a new prioritized SSP test suite is obtained. An experimental study using mutation testing analysis is conducted to assess the efficiency of the proposed test case prioritization method. The experimental study has demonstrated that prioritization approach can enhance the efficiency of SSP technique and achieved the purpose on this work.

Abstrak tesis yang dikemukakan kepada Universiti Putra Malaysia sebagai memenuhi keperluan untuk ijazah Sarjana Sains Komputer.

Meningkatkan Teknik *State-Sensitivity Partitioning* Menggunakan Pendekatan Keutamaan

Oleh

Nor Saadah binti Sulaiman

Julai 2015

Pengujian perisian adalah salah satu aktiviti dalam Jaminan Kualiti Perisian untuk mengukur kejayaan sesuatu projek. Dalam pengujian perisian dinamik, ujian kotak-kelabu telah dicadangkan untuk menimbangkan kedua dua pengujian kotak-putih (ujian berasaskan struktur) dan pengujian kotak hitam keselamatan (ujian berasaskan spesifikasi). *State-Sensitivity Partitioning* (SSP) adalah satu teknik baru di dalam ujian kotak-kelabu. Panjang jujukan ujian dengan pengulangan keadaan data menjadikan proses ujian yang mahal dan boleh membawa kepada ketidakcekapan. Oleh itu, kajian ini mencadangkan pendekatan keutamaan dalam menentukan dan menyusun semula kes-kes ujian SSP yang dijana dalam usaha untuk mempertingkat teknik SSP. Pertama, maklumat berdasarkan liputan iaitu liputan kod, liputan cawangan, liputan fungsi dan liputan laluan dikumpulkan. Kita menggunakan kaedah wajaran untuk mengira nilai keutamaan bagi setiap jujukan ujian berdasarkan maklumat liputan yang telah dikumpul. Kes-kes ujian kemudiannya disusun mengikut nilai keutamaan yang telah dikira maka dengan itu terhasil satu suit ujian yang baharu. Satu kajian eksperimen mengunakan analisis ujian mutasi dikendalikan untuk menilai kecekapan kaedah kes ujian keutamaan yang dicadangkan. Kajian eksperimen telah menunjukkan bahawa pendekatan keutamaan boleh meningkatkan kecekapan teknik SSP dan mencapai tujuan kajian ini.

TABLE OF CONTENT

ACKNOWLEDGEMENT

.

Alhamdulillah, all praises to Allah for giving me the strength and healthiness all of this time.

I would like to express my deepest gratitude to my supervisor, Dr. Salmi binti Baharom for providing me with an excellent guidance, encouragement and caring on doing this research. Her motivation and knowledge in experimental works is the most appreciated for me to go further in this field.

Special thanks to my moms, Pn. Hjh. Selamah binti Md. Lazi and Pn. Hjh. Maimunah binti Hidayat for all the prayers and motivation all the way through especially in some hardest time during this study.

To my other half, Mr. Mohd. Norhisham bin Musa and kids (Anissa and Irfan), your love support, time sacrifice and tolerance throughout my study are precious and priceless. Allah will bless us forever.

I also thank Public Service Department (PSD) and Malaysia Government for the scholarship support in my two years study. My gratitude also goes to my office supervisor; Pn. Vic, Pn. Rahamah, Pn. Zarina and Pn. Rohani for their encouragement, supportive and understanding my commitment to this study, not to forget also to my dearest Best Friend Forever and PPTM. I also thank Pn. Asmidar for patiently correcting my writing and Pn. Salwati for helping on tools for me.

In managing the tough time throughout this study, I have received helpful input and fully cooperation from my colleagues. Dear friends of Faculty of Computer Science and Information Technology (FSKTM), University Putra Malaysia batch 2013-2015, I appreciate every single assistance from you.

Finally, to all UPM staff, thank you for your facilitation. I would like to acknowledge any individual not mention here for his or her irreplaceable helps and cooperation.

DECLARATION

I declare that the thesis is my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously, and is not concurrently submitted for any other degree at University Putra Malaysia or at any other institutions.

NOR SAADAH BINTI SULAIMAN
DATE :

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATION

ABBRV.	DESCRIPTION
ABS	Absolute Value Insertion
AOR	Arithmetic Operator Replacement
APFD	Average of the Percentage of Fault Detected
API	Application Program Interface
BVA	Boundary Value Analysis
CP	Customer-Assigned Priority
E-R	Entity-Relationship
FED	Fault Exposing Potential
FIFO	First In First Out
FP	Fault Proneness
FSM	Finite State Machine
GRASP	Greedy Randomized Adaptive Search Procedure
IDE	Integrated Development Environment
IC	Implementation Complexity
JVM	Java Virtual Machine
LCR	Logical Connector Replacement
LOC	Line Of Code
MD	Module Documentation
MIDD	Module Internal Design Document
MIS	Module Interface Specification
PF	Prioritization Factor
PORT	Prioritization of Requirements for Test
RC	Requirements Complexity
ROR	Relational Operator Replacement
RV	Requirements Volatility
SSP	State-Sensitivity Partitioning
TC	Test Case
TS	Test Sequence
UOI	Unary Operator Insertion
WP	Weight Prioritization

CHAPTER I

INTRODUCTION

1.1 Research Background

Software Quality Assurance activities contain software testing as one of the critical elements to help the successfulness and sustainability of a project life cycle as it is the ultimate review of requirement specification, analysis, design and code implementation including maintenance phase.

In dynamic software testing, there are three different techniques which are Experience based, White-box testing and Black-box testing. White-box test case design technique is exercised within internal program logic and is structural testing. Black-box test case design technique is a specification based testing, conducted at software requirements or software interface. The intent in both cases is to find the maximum number of errors with the minimum amount of effort and time (Roger S. Pressman, 2001). However, in earlier studies (Baharom, 2008), grey-box testing approach has been suggested to consider both the external view and the internal structure of software under test (SUT) as an integrated approach.

State-Sensitivity-Partitioning (SSP) is a new technique to generate test cases for grey-box testing. The process is based on all-transition coverage criterion to avoid exhaustively testing the entire data states of a module. It will partition the test cases based on state's sensitivity towards events, conditions and action. The test data for this module testing is in a form of event sequences (or test sequence). There are six sequential steps to perform SSP (Baharom & Shukur, 2009).;

i. identifying sensitive access program,
ii. partitioning the states into equivalence classes,
iii. construct a state transition model,
iv. select test case based on all-transition coverage criteria,
v. add insensitive event at the end of each selected test cases and
vi. apply Boundary Value Analysis (BVA) technique to the input parameters.

(Akmam & Baharom, 2013) identified a problem that leads to inefficient time and ineffective process of testing in SSP. Test sequence generated randomly by SSP technique contains similar function either in the same test cases or other test cases thus it should be enhanced by eliminating redundant data states. This redundant data states will increase test suite size and leads to inefficient and ineffective way of testing process. The study proposed an SSP Optimizer Framework and a new technique to detect and eliminate redundant data states. Furthermore, executing all test cases will consume more time and infeasible especially in regression testing.

However, elimination of the test case might remove such important and significant data states that might be useful to detect fault. An early empirical study by (Rothermel, Harrold, Ostrin, & Hong, 1998) proposed the limitation of this approach is that it might reduce the ability to reveal faults in the software. The study also discovered that there is no close correlation between fault detection effectiveness and test suite size. There is limited confidence that the limited suite will be useful in detecting several other faults.

Therefore, this research will suggest another approach to optimize the original test case produced by SSP technique by using test case prioritization.

1.2 Research Motivation

There are no future investigation and empirical experiment in order to enhance the SSP technique, especially in test case prioritization. Since late 90's, the importance of prioritization topic is increasingly discussed and this field is still growing (Yoo & Harman, 2012). The main goal of prioritizing test case is to increase the effectiveness of fault detection and efficiency to rank the order of the test case with higher priority so that it will be executed early before the lower priority test case. In this case, testing will have maximum benefit even if it is prematurely halted in some point because faults are detected earlier.

1.3 Research Problem

A test sequence generated randomly by SSP might contain redundant data state. The lengthy sequence with redundant data state makes the testing process expensive and may lead to inefficiency. While elimination of the test case might remove important data state that can be useful in detecting several other faults, finding an appropriate approach might help to enhance the SSP technique and maximizing the benefit of testing.

1.4 Research Objectives

1.1 To enhance SSP technique by using test case prioritization technique.
1.2 To evaluate the efficiency of the enhanced approach.

1.5 Research Methodologies

Research methodology is the method of researcher intention used to collect data to solve a research problem. Research methodology is generally divided into quantitative and qualitative research. Both have its own purpose, process, data collection, data analysis and report findings that distinguish each other.

Qualitative research looks into qualities to examine complexities of a particular phenomenon with more variables of interests. The objective is to measure the variables in some way. Quantitative research, however measured collected data from an experiment to evaluate effectiveness variables of proposed approach. Correspond to some characteristics of quantitative research, Table 1.1 justify classification of this research as quantitative research:

Table 1.1: Characteristic of This Research

	Characteristic	Description
1	Purpose	To confirm and validate
2	Process	Focused, known variables, predetermined methods
3	Data Collection	Numeric data, representative, standardized instruments
4	Data analysis	Statistical analysis, deductive reasoning
5	Report finding	Numbers, statistics, aggregated data

In this research, the research design employed is an experimental design to quantify the efficiency of prioritization approach applied to SSP technique. Two different sets of test suites are compared. The first test suite is the original set generated randomly based on SSP technique. Another test suite is enhanced set after prioritizing the test suite based on some criteria that will be detailed later in Chapter V.

The efficiency of the approach is measured using a mutation-analysis technique. Mutation analysis or mutation testing is considered as a technique of systematically seeding faults into the system under test in order to create mutant versions. Both the test suites are then run on each of these versions. Prioritization approach rank the highest priority value of the test case on top of the test suites and the new enhanced test suite will have the most and earliest fault detection compared to the original test suite.

These research activities are divided into three phases; analysis and problem definition, design and development; and evaluation. The first activity is identifying the research problem. Research problem task includes defining research goals, identifying the research scope and available resources to plan and also ensuring feasibility of the research. A prototype is developed and validated based on prioritization approach that suits SPP technique. The evaluation phase is filled with experimental study and result from analysis to assess the efficiency of proposed approach. Overall research activities are depicted by Table 1.2.

Table 1.2: Research Activities

Phase	**Activities**	**Resources/Deliverables**
Analysis & Problem definition	Problem identification & analysis Identify relevant software & tools Proposed approach	Research problem, hypotheses and objectives Documents review Literature Review Subject Program Coverage information Eclipse <u>Deliverable</u> Proposal paper
Design and development	Design prototype Develop prototype Testing prototype	Weighted Method Formula Ms Excel MySQL Php <u>Deliverable</u> SSP Prioritization Tool
Evaluation	Experimental setup Experimental procedure Experimental study Data interpretation & analysis	JUnit test tool EclEmma Efficiency result Results analysis Influence Factor <u>Deliverables</u> Thesis

1.6 Research Contributions

Generally this research contributes towards efficiency of test case generated by SSP in order to detect fault earlier so that benefit of testing can be maximized.

Specific contributions in this research include:

1.6.1 Prioritization approach as an efficient way to enhance SSP technique.
1.6.2 Development of SSP Prioritization Tool to generate a list of prioritized test sequence using weighted method test case prioritization.
1.6.3 Empirical results on the efficiency of the approach regarding on fault detection in the modules.

1.7 Organization of Thesis

In this chapter, the motivation, research problem, research objectives and research contributions have been described. The remaining chapter of this thesis is organized as follows:

- CHAPTER II describes the comprehensive review of literature correlate with this research such as the SSP technique, test case prioritization, weighted method and mutation testing. These resources provide important information in undergoing this research as well as preparation of this thesis.

- CHAPTER III explains description of materials, theoretical approaches and experimental designs used to achieve objectives stated earlier in Chapter I. Materials and methods used in this research are also explained in detail and concisely for work extension in future.

- CHAPTER IV explained the detailed design of SSP Prioritization Tool combining with the implementation of the tool.

- CHAPTER V provides complete results on efficiency obtained in the research. Consequently, an analysis of the results is discussed and a conclusion is drawn at the end of the chapter.

- CHAPTER VI describes the overall significance of this research, thus a conclusion drawn in line with the objectives, concedes the limitations and suggests further research that may be carried out from this work.

1.8 Summary

This chapter describes the groundwork of the thesis which is research objectives and methodology used. The research problems and motivation prior to research objectives are explained and contributions of the research are briefly described. The outline of this thesis is stated at the end of this chapter.

REFERENCES

Akmam, M., & Baharom, S. (2013). Enhancing State-Sensitivity Partitioning Test Suite Using SSP Optimizer Technique, (December), 22–25.

Baharom, S. (2008). Module Documentation Based Testing Using Grey-Box Approach, (Md).

Baharom, S., & Shukur, Z. (2009). State-Sensitivity Partitioning Technique for Module Documentation-based Testing. *Business Transformation through Innovation and Knowledge Management: An Academic Perspective*, (Midd), 472–483.

Baharom, S., & Shukur, Z. (2011). An experimental assessment of module documentation-based testing. *Information and Software Technology*, *53*(7), 747–760. doi:10.1016/j.infsof.2011.01.005

Bradbury, J. S. (2007). Using Program Mutation for the Empirical Assessment of Fault Detection Techniques: A Comparison of Concurrency Testing and Model Checking. *PhD Dissertation*, 151.

Do, H., Member, S., & Rothermel, G. (2006). On the Use of Mutation Faults in Empirical Assessments of Test Case Prioritization Techniques, *32*(9), 733–752.

Do, H., & Rothermel, G. (2005). A Controlled Experiment Assessing Test Case Prioritization Techniques via Mutation Faults.

Duggal, G. (n.d.). Understanding regression testing techniques, (1).

Elbaum, S., & Rothermel, G. (2000). Prioritizing Test Cases for Regression Testing.

F W Struwig & G B Stead (2001), Planning, Reporting & Designing Research, 1-15

James F. Peters and Witold Pedrycs, “Software Engineering – An Engineering Approach”, John Wiley & Sons Inc. 2000, pp. 461-462.

Moore, I. (2001). Jester-a JUnit test tester. *Proc. of 2nd XP*, 84–87. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.25.1223&amp;rep=rep1&amp;type=pdf

Mungara, M. B. (2003). A Method for Systematically Generating Tests from Object-Oriented Class Interfaces A Method for Systematically Generating Tests From Object-Oriented Class Interfaces Mahesh Babu Mungara.

Prakash, N. (2013). Potentially Weighted Method for Test Case Prioritization, *18*, 7147–7156. doi:10.12733/jcis5860

Prakash, N., & Rangaswamy, T. R. (2013). Weighted Method For Coverage Based Test Case Prioritization, *56*(2), 235–243.

Roger S. Pressman. (2001). *Software Engineering A Practitioner ' S Approach*. Mc-Graw Hill.

Roongruangsuwan, S., & Daengdej, J. (2010). Test case prioritization techniques. *Journal of Theoretical and Applied Information Technology*, *18*(2), 45–60.

Rothermel, G., Harrold, M. J., Ostrin, J., & Hong, C. (1998). An Empirical Study of the Effects of Minimization on the Fault Detection Capabilities of Test Suites Test Suite Minimization Summary and Lit-. *Proceedings. International Conference on Software Maintenance (Cat. No. 98CB36272)*.

Rothermel, G., Untch, R. H., & Harrold, M. J. (1999). Test case prioritization: an empirical study. *Proceedings IEEE International Conference on Software Maintenance - 1999 (ICSM'99). "Software Maintenance for Business Change" (Cat. No.99CB36360)*, 179–188. doi:10.1109/ICSM.1999.792604

Sapaat, M. A., & Baharom, S. (2011). A Preliminary Investigation Towards Test Suite Optimization Approach for Enhanced State-Sensitivity Partitioning, (November), 4–9.

Srikanth, H., & Banerjee, S. (2012). Improving test efficiency through system test prioritization. *Journal of Systems and Software*, *85*(5), 1176–1187. doi:10.1016/j.jss.2012.01.007

Yoo, S., & Harman, M. (2012). Regression testing minimization , selection and prioritization : a survey, (March 2010), 67–120. doi:10.1002/stvr

http://eclipse.org

http://blog.takipi.com/we-analyzed-30000-github-projects-here-are-the-top-100-libraries-in-java-js-and-ruby/

http://pitest.org/java_mutation_testing_systems/

BIODATA OF STUDENT

This thesis is prepared by Nor Saadah binti Sulaiman (GS37979), in fulfilment of the requirement for degree of Master Computer Science at University Putra Malaysia. I am 2001 graduation of Bachelor Science (Hons.) Computer from University Teknologi Malaysia (UTM). Currently I am responsible as Information Technology Officer in one of the ministry at Putrajaya. I am pursuing my master degree as my part time and half-sponsored by Public Service Department (PSD), Malaysia.

Distribution of Statement Coverage

APPENDIX I - a

Line #	Code	TC	1	2	2	2	2	2	2	2	3	3	3	3	3	3	3	4	4	4	4	4	4	4	5	5	5	5	5	5
		TS	1	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6
1	public class Queue {																													
2	private static int QSIZE;																													
3	private int[] dataQ;																													
4	private int front;																													
5	private int rear;																													
6	private int len;																													
7	public Queue(int size){		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
8	QSIZE = size;		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
9	dataQ = new int[QSIZE];		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
10	front = 0;		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
11	rear = size - 1;		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
12	len = 0;		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
13	}		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
14	**public void add(int val) {**																													
15	if (len !=QSIZE){			1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
16	len++;			1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
17	rear = (rear + 1) % QSIZE;			1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
18	dataQ[rear] = val;			1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
19	}																													
20	}			1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
21	**public void remove() {**																													
22	if (len != 0) {		1								1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
23	len--;										1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
24	front = (front + 1) % QSIZE;										1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
25	}																													
26	}		1								1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
27	**public int front() {**																													
28	if (len == 0) {		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
29	return -1;		1																											
30	}else {																													
31	return dataQ[front];			1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
32	}																													
33	}																													
34	}																													
	SUM OF STATEMENT COVERED		**11**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**

Distribution of Statement Coverage

APPENDIX I - a

Line #	Code	TC	5	6	6	6	6	6	6	6	7	7	7	7	7	7	7	8	8	8	8	8	8	8	9	9	9	9	9	9
		TS	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6
1	public class Queue {																													
2	private static int QSIZE;																													
3	private int[] dataQ;																													
4	private int front;																													
5	private int rear;																													
6	private int len;																													
7	public Queue(int size){		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
8	QSIZE = size;		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
9	dataQ = new int[QSIZE];		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
10	front = 0;		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
11	rear = size - 1;		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
12	len = 0;		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
13	}		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
14	**public void add(int val) {**																													
15	if (len !=QSIZE){		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
16	len++;		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
17	rear = (rear + 1) % QSIZE;		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
18	dataQ[rear] = val;		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
19	}																													
20	}		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
21	**public void remove() {**																													
22	if (len != 0) {		1								1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
23	len--;		1								1	1	1	1	1	1	1	1	1	1	1	1	1	1						
24	front = (front + 1) % QSIZE;		1								1	1	1	1	1	1	1	1	1	1	1	1	1	1						
25	}																													
26	}		1								1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
27	**public int front() {**																													
28	if (len == 0) {		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
29	return -1;																													
30	}else {																													
31	return dataQ[front];		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
32	}																													
33	}																													
34	}																													
	SUM OF STATEMENT COVERED		**18**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**16**	**16**	**16**	**16**	**16**	**16**

Distribution of Statement Coverage

APPENDIX I - a

Line #	Code	TC	9	9	9	9	9	9	9	9	10	10	10	10	10	10	10	10
		TS	7	8	9	10	11	12	13	14	1	2	3	4	5	6	7	8
1	public class Queue {																	
2	private static int QSIZE;																	
3	private int[] dataQ;																	
4	private int front;																	
5	private int rear;																	
6	private int len;																	
7	public Queue(int size){		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
8	QSIZE = size;		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
9	dataQ = new int[QSIZE];		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
10	front = 0;		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
11	rear = size - 1;		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
12	len = 0;		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
13	}		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
14	**public void add(int val) {**																	
15	if (len !=QSIZE){		1	1	1	1	1	1	1	1		1	1	1	1	1	1	1
16	len++;		1	1	1	1	1	1	1	1		1	1	1	1	1	1	1
17	rear = (rear + 1) % QSIZE;		1	1	1	1	1	1	1	1		1	1	1	1	1	1	1
18	dataQ[rear] = val;		1	1	1	1	1	1	1	1		1	1	1	1	1	1	1
19	}																	
20	}		1	1	1	1	1	1	1	1		1	1	1	1	1	1	1
21	**public void remove() {**																	
22	if (len != 0) {		1								1	1	1	1	1	1	1	1
23	len--;											1	1	1	1	1	1	1
24	front = (front + 1) % QSIZE;											1	1	1	1	1	1	1
25	}																	
26	}		1								1	1	1	1	1	1	1	1
27	**public int front() {**																	
28	if (len == 0) {		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
29	return -1;										1							
30	}else {																	
31	return dataQ[front];		1	1	1	1	1	1	1	1		1	1	1	1	1	1	1
32	}																	
33	}																	
34	}																	
	SUM OF STATEMENT COVERED		**16**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**11**	**18**	**18**	**18**	**18**	**18**	**18**	**18**

Distribution of Function Coverage

APPENDIX I - b

Line #	Code																													
		TC	1	2	2	2	2	2	2	2	3	3	3	3	3	3	3	4	4	4	4	4	4	4	5	5	5	5	5	5
		TS	1	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6
1	public class Queue {																													
2	private static int QSIZE;																													
3	private int[] dataQ;																													
4	private int front;																													
5	private int rear;																													
6	private int len;																													
7	public Queue(int size){		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
8	QSIZE = size;																													
9	dataQ = new int[QSIZE];																													
10	front = 0;																													
11	rear = size - 1;																													
12	len = 0;																													
13	}																													
14	**public void add(int val) {**			1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
15	if (len !=QSIZE){																													
16	len++;																													
17	rear = (rear + 1) % QSIZE;																													
18	dataQ[rear] = val;																													
19	}																													
20	}																													
21	**public void remove() {**		1								1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
22	if (len != 0) {																													
23	len--;																													
24	front = (front + 1) % QSIZE;																													
25	}																													
26	}																													
27	**public int front() {**		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
28	if (len == 0) {																													
29	return -1;																													
30	}else {																													
31	return dataQ[front];																													
32	}																													
33	}																													
34	}																													
	SUM OF FUNCTION COVERED		**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**

Distribution of Function Coverage

APPENDIX I - b

Line #	Code	TC	5	6	6	6	6	6	6	6	7	7	7	7	7	7	7	8	8	8	8	8	8	8	9	9	9	9	9	9
		TS	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6
1	public class Queue {																													
2	private static int QSIZE;																													
3	private int[] dataQ;																													
4	private int front;																													
5	private int rear;																													
6	private int len;																													
7	public Queue(int size){		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
8	QSIZE = size;																													
9	dataQ = new int[QSIZE];																													
10	front = 0;																													
11	rear = size - 1;																													
12	len = 0;																													
13	}																													
14	**public void add(int val) {**		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
15	if (len !=QSIZE){																													
16	len++;																													
17	rear = (rear + 1) % QSIZE;																													
18	dataQ[rear] = val;																													
19	}																													
20	}																													
21	**public void remove() {**		1								1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
22	if (len != 0) {																													
23	len--;																													
24	front = (front + 1) % QSIZE;																													
25	}																													
26	}																													
27	**public int front() {**		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
28	if (len == 0) {																													
29	return -1;																													
30	}else {																													
31	return dataQ[front];																													
32	}																													
33	}																													
34	}																													
	SUM OF FUNCTION COVERED		**4**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**

Distribution of Function Coverage

APPENDIX I - b

Line #	Code	TC	9	9	9	9	9	9	9	9	10	10	10	10	10	10	10	10
		TS	7	8	9	10	11	12	13	14	1	2	3	4	5	6	7	8
1	public class Queue {																	
2	private static int QSIZE;																	
3	private int[] dataQ;																	
4	private int front;																	
5	private int rear;																	
6	private int len;																	
7	public Queue(int size){		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
8	QSIZE = size;																	
9	dataQ = new int[QSIZE];																	
10	front = 0;																	
11	rear = size - 1;																	
12	len = 0;																	
13	}																	
14	**public void add(int val) {**		1	1	1	1	1	1	1	1		1	1	1	1	1	1	1
15	if (len !=QSIZE){																	
16	len++;																	
17	rear = (rear + 1) % QSIZE;																	
18	dataQ[rear] = val;																	
19	}																	
20	}																	
21	**public void remove() {**		1								1	1	1	1	1	1	1	1
22	if (len != 0) {																	
23	len--;																	
24	front = (front + 1) % QSIZE;																	
25	}																	
26	}																	
27	**public int front() {**		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
28	if (len == 0) {																	
29	return -1;																	
30	}else {																	
31	return dataQ[front];																	
32	}																	
33	}																	
34	}																	
	SUM OF FUNCTION COVERED		**4**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**4**	**4**	**4**	**4**	**4**	**4**	**4**

Distribution of Path Coverage

APPENDIX I - c

Line #	Code																													
		TC	1	2	2	2	2	2	2	2	3	3	3	3	3	3	3	4	4	4	4	4	4	4	5	5	5	5	5	5
		TS	1	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6
1	public class Queue {																													
2	private static int QSIZE;																													
3	private int[] dataQ;																													
4	private int front;																													
5	private int rear;																													
6	private int len;																													
7	public Queue(int size){																													
8	QSIZE = size;																													
9	dataQ = new int[QSIZE];																													
10	front = 0;																													
11	rear = size - 1;																													
12	len = 0;																													
13	}																													
14	**public void add(int val) {**																													
15	if (len !=QSIZE){																													
16	len++;		1	1	1	1	1	1	1		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
17	rear = (rear + 1) % QSIZE;																													
18	dataQ[rear] = val;																													
19	}																													
20	}										1	1	1	1	1	1	1													
21	**public void remove() {**																													
22	if (len != 0) {																													
23	len--;										1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
24	front = (front + 1) % QSIZE;																													
25	}																													
26	}									1																				
27	**public int front() {**																													
28	if (len == 0) {																													
29	return -1;									1																				
30	}else {																													
31	return dataQ[front];		1	1	1	1	1	1	1		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
32	}																													
33	}																													
34	}																													
	SUM OF PATH COVERED		**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**

Distribution of Path Coverage

APPENDIX I - c

Line #	Code	TC	5	6	6	6	6	6	6	6	7	7	7	7	7	7	7	8	8	8	8	8	8	8	9	9	9	9	9	9
		TS	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6
1	public class Queue {																													
2	private static int QSIZE;																													
3	private int[] dataQ;																													
4	private int front;																													
5	private int rear;																													
6	private int len;																													
7	public Queue(int size){																													
8	QSIZE = size;																													
9	dataQ = new int[QSIZE];																													
10	front = 0;																													
11	rear = size - 1;																													
12	len = 0;																													
13	}																													
14	**public void add(int val) {**																													
15	if (len !=QSIZE){																													
16	len++;		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
17	rear = (rear + 1) % QSIZE;																													
18	dataQ[rear] = val;																													
19	}																													
20	}																													
21	**public void remove() {**																													
22	if (len != 0) {																													
23	len--;		1								1	1	1	1	1	1	1	1	1	1	1	1	1	1						
24	front = (front + 1) % QSIZE;																													
25	}																													
26	}																								1	1	1	1	1	1
27	**public int front() {**																													
28	if (len == 0) {																													
29	return -1;																													
30	}else {																													
31	return dataQ[front];		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
32	}																													
33	}																													
34	}																													
	SUM OF PATH COVERED		**3**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**

Distribution of Path Coverage

APPENDIX I - c

Line #	Code	TC	9	9	9	9	9	9	9	9	10	10	10	10	10	10	10	10
		TS	7	8	9	10	11	12	13	14	1	2	3	4	5	6	7	8
1	public class Queue {																	
2	private static int QSIZE;																	
3	private int[] dataQ;																	
4	private int front;																	
5	private int rear;																	
6	private int len;																	
7	public Queue(int size){																	
8	QSIZE = size;																	
9	dataQ = new int[QSIZE];																	
10	front = 0;																	
11	rear = size - 1;																	
12	len = 0;																	
13	}																	
14	**public void add(int val) {**																	
15	if (len !=QSIZE){																	
16	len++;		1	1	1	1	1	1	1	1		1	1	1	1	1	1	1
17	rear = (rear + 1) % QSIZE;																	
18	dataQ[rear] = val;																	
19	}																	
20	}			1	1	1	1	1	1	1								
21	**public void remove() {**																	
22	if (len != 0) {																	
23	len--;											1	1	1	1	1	1	1
24	front = (front + 1) % QSIZE;																	
25	}																	
26	}		1								1	1	1	1	1	1	1	1
27	**public int front() {**																	
28	if (len == 0) {																	
29	return -1;											1	1	1	1	1	1	1
30	}else {																	
31	return dataQ[front];		1	1	1	1	1	1	1	1								
32	}																	
33	}										1							
34	}																	
	SUM OF PATH COVERED		**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**2**	**4**	**4**	**4**	**4**	**4**	**4**	**4**

Distribution of Branch Coverage

APPENDIX I - d

Line #	Code	TC	1	2	2	2	2	2	2	2	3	3	3	3	3	3	3	4	4	4	4	4	4	4	5	5	5	5	5	5
		TS	1	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6
1	public class Queue {																													
2	private static int QSIZE;																													
3	private int[] dataQ;																													
4	private int front;																													
5	private int rear;																													
6	private int len;																													
7	public Queue(int size){																													
8	QSIZE = size;																													
9	dataQ = new int[QSIZE];																													
10	front = 0;																													
11	rear = size - 1;																													
12	len = 0;																													
13	}																													
14	**public void add(int val) {**																													
15	if (len !=QSIZE){																													
16	len++;		1	1	1	1	1	1	1		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
17	rear = (rear + 1) % QSIZE;																													
18	dataQ[rear] = val;																													
19	}																													
20	}										1	1	1	1	1	1	1													
21	**public void remove() {**																													
22	if (len != 0) {																													
23	len--;										1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
24	front = (front + 1) % QSIZE;																													
25	}																													
26	}									1																				
27	**public int front() {**																													
28	if (len == 0) {																													
29	return -1;									1																				
30	}else {																													
31	return dataQ[front];		1	1	1	1	1	1	1		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
32	}																													
33	}																													
34	}																													
	SUM OF BRANCH COVERED		**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**

Distribution of Branch Coverage

APPENDIX I - d

Line #	Code	TC	5	6	6	6	6	6	6	6	7	7	7	7	7	7	7	8	8	8	8	8	8	8	9	9	9	9	9	9
		TS	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6
1	public class Queue {																													
2	private static int QSIZE;																													
3	private int[] dataQ;																													
4	private int front;																													
5	private int rear;																													
6	private int len;																													
7	public Queue(int size){																													
8	QSIZE = size;																													
9	dataQ = new int[QSIZE];																													
10	front = 0;																													
11	rear = size - 1;																													
12	len = 0;																													
13	}																													
14	**public void add(int val) {**																													
15	if (len !=QSIZE){																													
16	len++;		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
17	rear = (rear + 1) % QSIZE;																													
18	dataQ[rear] = val;																													
19	}																													
20	}																													
21	**public void remove() {**																													
22	if (len != 0) {																													
23	len--;		1								1	1	1	1	1	1	1	1	1	1	1	1	1	1						
24	front = (front + 1) % QSIZE;																													
25	}																													
26	}																								1	1	1	1	1	1
27	**public int front() {**																													
28	if (len == 0) {																													
29	return -1;																													
30	}else {																													
31	return dataQ[front];		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
32	}																													
33	}																													
34	}																													
	SUM OF BRANCH COVERED		**3**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**

Distribution of Branch Coverage

APPENDIX I - d

Line #	Code	TC	9	9	9	9	9	9	9	9	10	10	10	10	10	10	10	10
		TS	7	8	9	10	11	12	13	14	1	2	3	4	5	6	7	8
1	public class Queue {																	
2	private static int QSIZE;																	
3	private int[] dataQ;																	
4	private int front;																	
5	private int rear;																	
6	private int len;																	
7	public Queue(int size){																	
8	QSIZE = size;																	
9	dataQ = new int[QSIZE];																	
10	front = 0;																	
11	rear = size - 1;																	
12	len = 0;																	
13	}																	
14	**public void add(int val) {**																	
15	if (len !=QSIZE){																	
16	len++;		1	1	1	1	1	1	1	1		1	1	1	1	1	1	1
17	rear = (rear + 1) % QSIZE;																	
18	dataQ[rear] = val;																	
19	}																	
20	}			1	1	1	1	1	1	1								
21	**public void remove() {**																	
22	if (len != 0) {																	
23	len--;											1	1	1	1	1	1	1
24	front = (front + 1) % QSIZE;																	
25	}																	
26	}		1								1	1	1	1	1	1	1	1
27	**public int front() {**																	
28	if (len == 0) {																	
29	return -1;											1	1	1	1	1	1	1
30	}else {																	
31	return dataQ[front];		1	1	1	1	1	1	1	1								
32	}																	
33	}										1							
34	}																	
	SUM OF BRANCH COVERED		**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**2**	**4**	**4**	**4**	**4**	**4**	**4**	**4**

Appendix II - a

```
package com.queue.report;

public class MyCQreport {
        private static int QSIZE;
        private int[] dataQ;
        private int front;
        private int rear;
        private int len;
        private int _rear;
        private int _len;
        private int _front;
        private int[] _dataQ;

public MyCQreport(int size){
        QSIZE = size;
        dataQ = new int[QSIZE];
        front = 0;
        rear = size - 1;
        len = 0;
        _front = front;
        _rear = rear;
        _len = len;
        _dataQ = dataQ;
        }

        public void add(int val) {
                if (len !=QSIZE){ // if not full
                        System.out.println("\nAdd: Function-ADD passed.");
                        System.out.println("\tAdd: Branch 1 : len <> QSIZE.");
                        System.out.println("\t\tAdd: Statement 1 : if Len before =
                        " + len);
                        len++;
                        System.out.println("\t\tAdd: Statement 2 : Len++ after = "
                        + len);
                        rear = (rear + 1) % QSIZE;
                        System.out.println("\t\tAdd: Statement 3 : Rear+1 = " +
                        rear);
                        dataQ[rear] = val;
                        System.out.println("\t\tAdd: Statement 4: dataQ[" + rear +
                        "] = " + val);
                }
                else
                        System.out.println("\tAdd: Branch 2 : len == QSIZE. Queue
                        is full.");
                }

        public void remove() {
                if (len != 0) { // if not empty
                        System.out.println("\nRmv: Function-REMOVE passed.");
                        System.out.println("\tRmv: Branch 1 : len <> 0.");
                        System.out.println("\t\tRmv: Statement 1 : if Len before =
                        " + len);
                        len--;
                        System.out.println("\t\tRmv: Statement 2 : Len-- after = "
                        + len);
                        front = (front + 1) % QSIZE;
                        System.out.println("\t\tRmv: Statement 3 : Front+1 = " +
                        front);
                }
                System.out.println("\tRmv: Branch 2 len == 0. Queue is empty.");
        }

        public int front() {
                if (len == 0) {
                        System.out.println("\nFrt: Function-FRONT passed.");
                        System.out.println("\tFrt: Branch 1 len == 0.");
                        System.out.println("\t\tFrt: Statement 1 if : Return -1");
                        return -1;
                        }else {
                                System.out.println("\tFrt: Branch 2 len <> 0.");
                                return dataQ[front];
                        }
        }

             public void printq() {
                System.out.println();
                System.out.println(":: Front: " + front + ", Rear: " + rear + ",
                Len: " + len);
                 System.out.print(":: q: ");
                 for (int i = 0; i < QSIZE; i++) {
                     System.out.print("q[" + i + "]=" + dataQ[i] + "; ");
                 }
                 System.out.println();
             }

        public int getQSIZE() {
                return QSIZE;
        }

        public int get_Rear() {
                return _rear;
        }

        public int getRear() {
                return rear;
        }

        public int get_Len() {
                return _len;
        }

        public int getLen() {
                return len;
        }

        public int get_Front() {
                return _front;
        }

        public int getFront() {
                return front;
        }

        public int[] get_DataQ() {
                return _dataQ;
```

```
	}
	public int[] getDataQ() {
		return dataQ;
	}
	public void update() {
		_front = front;
		_rear = rear;
		_len = len;
		_dataQ = dataQ;
	}

	public static void main (String [] args) {

	int qSize = 10;
	MyCQreport q = new MyCQreport(qSize);

	System.out.println("Test Case 5");
	q.add(0);
	q.printq();
	q.add(1);
	q.printq();
	q.remove();
	q.printq();
	q.front();
	q.printq();
	}
}
```

Appendix II - b

```
Test Case 5

Add: Function-ADD passed.
      Add: Branch 1 : len <> QSIZE.
            Add: Statement 1 : if Len before = 0
            Add: Statement 2 : Len++ after = 1
            Add: Statement 3 : Rear+1 = 0
            Add: Statement 4 : dataQ[0] = 0

:: Front: 0, Rear: 0, Len: 1
:: q: q[0]=0; q[1]=0; q[2]=0; q[3]=0; q[4]=0; q[5]=0; q[6]=0; q[7]=0; q[8]=0;
q[9]=0;

Add: Function-ADD passed.
      Add: Branch 1 : len <> QSIZE.
            Add: Statement 1 : if Len before = 1
            Add: Statement 2 : Len++ after = 2
            Add: Statement 3 : Rear+1 = 1
            Add: Statement 4 : dataQ[1] = 1

:: Front: 0, Rear: 1, Len: 2
:: q: q[0]=0; q[1]=1; q[2]=0; q[3]=0; q[4]=0; q[5]=0; q[6]=0; q[7]=0; q[8]=0;
q[9]=0;

Rmv: Function-REMOVE passed.
      Rmv: Branch 1 : len <> 0.
            Rmv: Statement 1 : if Len before = 2
            Rmv: Statement 2 : Len-- after = 1
            Rmv: Statement 3 : Front+1 = 1
      Rmv: Branch 2 len == 0. Queue is empty.

:: Front: 1, Rear: 1, Len: 1
:: q: q[0]=0; q[1]=1; q[2]=0; q[3]=0; q[4]=0; q[5]=0; q[6]=0; q[7]=0; q[8]=0;
q[9]=0;
      Frt: Branch 2 len <> 0.

:: Front: 1, Rear: 1, Len: 1
:: q: q[0]=0; q[1]=1; q[2]=0; q[3]=0; q[4]=0; q[5]=0; q[6]=0; q[7]=0; q[8]=0;
q[9]=0;
```

APPENDIX III - a

Test Case	Test Sequence	Test cases and its associated codes, functions, paths and branches				Test cases and its Criteria Value of codes, functions, paths and branches				
		Table 1				Table 2				
		Code Coverage	Function coverage	Path Coverage	Branch Coverage	Code Coverage Criteria Value	Function Coverage Criteria Value	Path Coverage Criteria Value	Branch Coverage Criteria Value	Test Case Priority Value
1	1	15	3	2	2	6.82	7.5	5	5	6.08
2	1	18	3	2	2	8.18	7.5	5	5	6.42
2	2	18	3	2	2	8.18	7.5	5	5	6.42
2	3	18	3	2	2	8.18	7.5	5	5	6.42
2	4	18	3	2	2	8.18	7.5	5	5	6.42
2	5	18	3	2	2	8.18	7.5	5	5	6.42
2	6	18	3	2	2	8.18	7.5	5	5	6.42
2	7	18	3	2	2	8.18	7.5	5	5	6.42
3	1	22	4	3	3	10.00	10	7.5	7.5	8.75
3	2	22	4	3	3	10.00	10	7.5	7.5	8.75
3	3	22	4	3	3	10.00	10	7.5	7.5	8.75
3	4	22	4	3	3	10.00	10	7.5	7.5	8.75
3	5	22	4	3	3	10.00	10	7.5	7.5	8.75
3	6	22	4	3	3	10.00	10	7.5	7.5	8.75
3	7	22	4	3	3	10.00	10	7.5	7.5	8.75
4	1	22	4	3	3	10.00	10	7.5	7.5	8.75
4	2	22	4	3	3	10.00	10	7.5	7.5	8.75
4	3	22	4	3	3	10.00	10	7.5	7.5	8.75
4	4	22	4	3	3	10.00	10	7.5	7.5	8.75
4	5	22	4	3	3	10.00	10	7.5	7.5	8.75
4	6	22	4	3	3	10.00	10	7.5	7.5	8.75
4	7	22	4	3	3	10.00	10	7.5	7.5	8.75
5	1	22	4	3	3	10.00	10	7.5	7.5	8.75
5	2	22	4	3	3	10.00	10	7.5	7.5	8.75
5	3	22	4	3	3	10.00	10	7.5	7.5	8.75
5	4	22	4	3	3	10.00	10	7.5	7.5	8.75
5	5	22	4	3	3	10.00	10	7.5	7.5	8.75
5	6	22	4	3	3	10.00	10	7.5	7.5	8.75
5	7	22	4	3	3	10.00	10	7.5	7.5	8.75
6	1	18	3	2	2	8.18	7.5	5	5	6.42
6	2	18	3	2	2	8.18	7.5	5	5	6.42
6	3	18	3	2	2	8.18	7.5	5	5	6.42
6	4	18	3	2	2	8.18	7.5	5	5	6.42
6	5	18	3	2	2	8.18	7.5	5	5	6.42
6	6	18	3	2	2	8.18	7.5	5	5	6.42
6	7	18	3	2	2	8.18	7.5	5	5	6.42

7	1	22	4	3	3	10.00	10	7.5	7.5	8.75
7	2	22	4	3	3	10.00	10	7.5	7.5	8.75
7	3	22	4	3	3	10.00	10	7.5	7.5	8.75
7	4	22	4	3	3	10.00	10	7.5	7.5	8.75
7	5	22	4	3	3	10.00	10	7.5	7.5	8.75
7	6	22	4	3	3	10.00	10	7.5	7.5	8.75
7	7	22	4	3	3	10.00	10	7.5	7.5	8.75
8	1	22	4	4	4	10.00	10	10	10	10.00
8	2	22	4	4	4	10.00	10	10	10	10.00
8	3	22	4	4	4	10.00	10	10	10	10.00
8	4	22	4	4	4	10.00	10	10	10	10.00
8	5	22	4	4	4	10.00	10	10	10	10.00
8	6	22	4	4	4	10.00	10	10	10	10.00
8	7	22	4	4	4	10.00	10	10	10	10.00
9	1	20	4	3	3	9.09	10	7.5	7.5	8.52
9	2	20	4	3	3	9.09	10	7.5	7.5	8.52
9	3	20	4	3	3	9.09	10	7.5	7.5	8.52
9	4	20	4	3	3	9.09	10	7.5	7.5	8.52
9	5	20	4	3	3	9.09	10	7.5	7.5	8.52
9	6	20	4	3	3	9.09	10	7.5	7.5	8.52
9	7	20	4	3	3	9.09	10	7.5	7.5	8.52
9	8	18	3	3	3	8.18	7.5	7.5	7.5	7.67
9	9	18	3	3	3	8.18	7.5	7.5	7.5	7.67
9	10	18	3	3	3	8.18	7.5	7.5	7.5	7.67
9	11	18	3	3	3	8.18	7.5	7.5	7.5	7.67
9	12	18	3	3	3	8.18	7.5	7.5	7.5	7.67
9	13	18	3	3	3	8.18	7.5	7.5	7.5	7.67
9	14	18	3	3	3	8.18	7.5	7.5	7.5	7.67
10	1	15	3	2	2	6.82	7.5	5	5	6.08
10	2	22	4	4	4	10.00	10	10	10	10.00
10	3	22	4	4	4	10.00	10	10	10	10.00
10	4	22	4	4	4	10.00	10	10	10	10.00
10	5	22	4	4	4	10.00	10	10	10	10.00
10	6	22	4	4	4	10.00	10	10	10	10.00
10	7	22	4	4	4	10.00	10	10	10	10.00
10	8	22	4	4	4	10.00	10	10	10	10.00

Appendix IV - a

home.php

```
<html lang="en">
<head>
  <meta charset="utf-8">
    <title>SSP Test Case Prioritization</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="">
    <meta name="author" content="">
    <!-- Le styles -->
    <link href="\bootstrap-3.3.5-dist\css\bootstrap.css" rel="stylesheet">
    <link href="\bootstrap-3.3.5-dist\css\justified-nav.css" rel="stylesheet">
    <link href="css/style2.css" rel="stylesheet" type="text/css" />
    <link href="css/style.css" rel="stylesheet" type="text/css" />
</head>

  <body>
    <div class="container">
      <!-- The justified navigation menu is meant for single line per list item.
           Multiple lines will require custom code not provided by Bootstrap. -->
      <div class="masthead">
        <h3 class="text-muted" align="center">Enhancing State-Sensitivity
        Partitioning by using Prioritization Approach</h3>
        <nav>
          <ul class="nav nav-justified">
                <li class="active"><a href="#">Home</a></li>
                <li><a href="\tcpssp\form.php">Upload</a></li>
                <li><a href="\tcpssp\input.php">Output</a></li>
                <li><a href="#">Criteria Value</a></li>
                <li><a href="#">Prioritized List</a></li>
                <li><a href="#">About</a></li>
          </ul>
        </nav>
      </div>

  <div class="jumbotron">
        <h2>Introduction</h2>
        <p class="lead">This application is meant to generate Prioritized SSP Test
        Sequence list. Click <a href="\tcpssp\form.php">Upload</a> page to upload
        new file</p>
  </div>
  <br/>
  <br/>
  </body>
</html>
```

Appendix IV - b

form.php

```
<html lang="en">
<head>
  <meta charset="utf-8">
    <title>SSP Test Case Prioritization</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="">
    <meta name="author" content="">
    <!-- Le styles -->
    <link href="\bootstrap-3.3.5-dist\css\bootstrap.css" rel="stylesheet">
    <link href="\bootstrap-3.3.5-dist\css\justified-nav.css" rel="stylesheet">
    <link href="css/style2.css" rel="stylesheet" type="text/css" />
    <link href="css/style.css" rel="stylesheet" type="text/css" />
</head>
  <body>
    <div class="container">
      <div class="masthead">
        <h3 class="text-muted" align="center">Enhancing State-Sensitivity
        Partitioning by using Prioritization Approach</h3>
        <nav>
          <ul class="nav nav-justified">
                <li><a href=""\tcpssp\home.php"">Home</a></li>
                <li class="active"><a href="#">Upload</a></li>
                <li><a href="#">Output</a></li>
                <li><a href="#">Criteria Value</a></li>
                <li><a href="#">Prioritized List</a></li>
                <li><a href="#">About</a></li>
          </ul>
        </nav>
      </div>

    <div class="jumbotron">
        <h2>Upload</h2>
        <p class="lead">Upload SSP Test Suite in .csv format. Choose file and
        click Upload and Tabulate.</p>
    </div><br/>

    <form method="post" action="upload.php" enctype="multipart/form-data">
      <table width="60%" align=center class="tableContent">
        <th colspan="2">Upload File</th>
        <tr>
          <td>File : </td><td><input type="file" name="fileName"></td>
        </tr>
        <tr>
          <td>Description : </td><td><textarea cols="50" rows="5"
           name="fileDesc"></textarea></td>
        </tr>
        <tr>
          <td></td><td><input type="submit" name="upload" value="     Upload &
           Tabulate     " class="btn btn-large btn-success" /></td>
        </tr>
      </table>
    </form>
  </body>
</html>
```

upload.php

```
<?php
include('conn.php');

$convertFunctionPath = 'input.php';

//uploaded file extension
$uploadedFileSplit = explode('.',$_FILES['fileName']['name']);
$uploadedFileName = $uploadedFileSplit[0];
$uploadedFileExt = $uploadedFileSplit[1];

//filename to be store
$uploadfile = $uploadedFileName.date('YmdHis').rand(1,999).'.'.$uploadedFileExt;

//This gets all the other information from the form
$fileDesc = $_POST['fileDesc'];
$fileName = basename($uploadfile);
$tmpName  = $_FILES['fileName']['tmp_name'];
$fileSize = $_FILES['fileName']['size'];
$fileType = $_FILES['fileName']['type'];

//This is the directory where images will be saved
$target = "upload/";
$target = $target . $fileName;

//Writes the Filename to the server
if(move_uploaded_file($tmpName, $target)) {

  $insertDoc =
    "INSERT INTO project_master.DOCUMENT(
      DOC_NAME, DOC_FILE, DOC_TYPE, DOC_SIZE
      ) VALUES(
      '$fileDesc', '$fileName', '$fileType', '$fileSize'
    )";
  $insertDocRs = mysql_query($insertDoc, $con) or die(mysql_error());

  $getDocId = "SELECT IFNULL(MAX(DOC_ID),0) AS MAX_DOC_ID FROM
   project_master.DOCUMENT";
  $getDocIdRs = mysql_query($getDocId, $con) or die(mysql_error());

  while($dataArray = mysql_fetch_array($getDocIdRs)) {
    $maxDocId = $dataArray['MAX_DOC_ID'];
  }

} else {
    //Gives and error if its not
    echo "Sorry, there was a problem uploading your file.";
}

//close connection
mysql_close($con);
?>
```

Appendix IV - c

input.php

```
<html lang="en">
<head>
  <meta charset="utf-8">
    <title>SSP Test Case Prioritization</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="">
    <meta name="author" content="">
    <!-- Le styles -->
    <link href="\bootstrap-3.3.5-dist\css\bootstrap.css" rel="stylesheet">
    <link href="\bootstrap-3.3.5-dist\css\justified-nav.css" rel="stylesheet">
    <link href="css/style2.css" rel="stylesheet" type="text/css" />
    <link href="css/style.css" rel="stylesheet" type="text/css" />
</head>
  <body>
    <div class="container">
      <div class="masthead">
        <h3 class="text-muted" align="center">Enhancing State-Sensitivity
        Partitioning by using Prioritization Approach</h3>
        <nav>
          <ul class="nav nav-justified">
                <li><a href="\tcpssp\home.php">Home</a></li>
                <li><a href="\tcpssp\form.php">Upload</a></li>
                <li class="active"><a href="#">Output</a></li>
                <li><a href="#">Criteria Value</a></li>
                <li><a href="#">Prioritized List</a></li>
                <li><a href="#">About</a></li>
          </ul>
        </nav>
      </div>
      <div class="jumbotron">
        <h2>Test Case and Test Sequence List</h2>
        <p class="lead">This table shows Test Case and Test Sequence List with
        their associated codes, functions, paths and branches from the input
        uploaded before. At the end of the table is maximum value of each
        Coverage Criterion.</p>
      </div>
<br/>
<?php

include('conn.php');
include('func.php');

$maxDocId = $_GET['docId'];

/**
 * The Logic is here
 */
$getCountRec =
  "SELECT COUNT(*) AS TOTAL_REC FROM project_master.PRIORITIZED_LIST WHERE DOC_ID
= '$maxDocId'";
$getCountRecRs = mysql_query($getCountRec, $con) or die(mysql_error());

while($dataArray = mysql_fetch_array($getCountRecRs)) {
  $totalRec = $dataArray['TOTAL_REC'];
}
```

```
$getDocInfo =
  "SELECT DOC_NAME, DOC_FILE FROM project_master.DOCUMENT WHERE DOC_ID =
'$maxDocId'";
$getDocInfoRs = mysql_query($getDocInfo, $con) or die(mysql_error());

while($dataArray = mysql_fetch_array($getDocInfoRs)) {
  $docDesc = $dataArray['DOC_NAME'];
  $docPath = 'upload/'.$dataArray['DOC_FILE'];
}

//Read input from file in .csv format
$arrayInput = csv_to_array($docPath);
$arrayInputCount = count($arrayInput);

//$inputCC = array();
for ($x=0; $x<$arrayInputCount; $x++) {
  $inputCC[] = $arrayInput[$x]['CC'];
  $inputFC[] = $arrayInput[$x]['FC'];
  $inputPC[] = $arrayInput[$x]['PC'];
  $inputBC[] = $arrayInput[$x]['BC'];
  $inputDS[] = $arrayInput[$x]['DS'];
}

$maxCC = max($inputCC);
$maxFC = max($inputFC);
$maxPC = max($inputPC);
$maxBC = max($inputBC);
?>
<br/>

<table class="table table-hover" border="1" cellspacing="0" cellpadding="3"
width="100%">
  <thead class="alert-info">
    <tr>
      <th rowspan="2">Test Case</th><th rowspan="2">Test Sequence</th>
    <th colspan="4">Test cases and its associated codes, functions, paths and
    branches</th>
    </tr>
    <tr>
      <th>Code Coverage</th><th>Function coverage</th><th>Path
       Coverage</th><th>Branch Coverage</th>
    </tr>
  </thead>

  <?php
  for ($y=0; $y<$arrayInputCount; $y++) {

  $resultCC[] = sprintf ("%.2f", $arrayInput[$y]['CC']/$maxCC*10);
  $resultFC[] = sprintf ("%.2f", $arrayInput[$y]['FC']/$maxFC*10);
  $resultPC[] = sprintf ("%.2f", $arrayInput[$y]['PC']/$maxPC*10);
  $resultBC[] = sprintf ("%.2f", $arrayInput[$y]['BC']/$maxBC*10);

  $calcValueTCP = $resultCC[$y]+$resultFC[$y]+$resultPC[$y]+$resultBC[$y];
  $resultTCP[] = sprintf ("%.2f", $calcValueTCP/4);

  if($totalRec == 0) {
    $insertValue =
      "INSERT INTO project_master.PRIORITIZED_LIST(
        DOC_ID, TC, TS, INPUT_CC, INPUT_FC, INPUT_PC, INPUT_BC,
        CRITERIA_VAL_CC, CRITERIA_VAL_FC, CRITERIA_VAL_PC, CRITERIA_VAL_BC,
        VAL_TCP, DATA_STATE
      ) VALUES (
        ".$maxDocId.",'".$arrayInput[$y]['TC']."', '".$arrayInput[$y]['TS']."',
        ".$arrayInput[$y]['CC'].", ".$arrayInput[$y]['FC'].",
        ".$arrayInput[$y]['PC'].", ".$arrayInput[$y]['BC'].",
        ".$resultCC[$y].", ".$resultFC[$y].", ".$resultPC[$y].",
        ".$resultBC[$y].",
        ".$resultTCP[$y].",'".$arrayInput[$y]['DS']."'
      )";
    $insertValueRs = mysql_query($insertValue, $con) or die(mysql_error());
  }

  if($arrayInput[$y]['TC'] == '1') $style1 = "bg01";
  if($arrayInput[$y]['TC'] == '2') $style1 = "bg02";
  if($arrayInput[$y]['TC'] == '3') $style1 = "bg03";
  if($arrayInput[$y]['TC'] == '4') $style1 = "bg04";
  if($arrayInput[$y]['TC'] == '5') $style1 = "bg05";
  if($arrayInput[$y]['TC'] == '6') $style1 = "bg06";
  if($arrayInput[$y]['TC'] == '7') $style1 = "bg07";
  if($arrayInput[$y]['TC'] == '8') $style1 = "bg08";
  if($arrayInput[$y]['TC'] == '9') $style1 = "bg09";
  if($arrayInput[$y]['TC'] == '10') $style1 = "bg10";

  ?>
  <tr class=<?php echo $style1; ?>>
    <td><?php echo $arrayInput[$y]['TC']; ?></td>
    <td><?php echo $arrayInput[$y]['TS']; ?></td>
    <td align="right"><?php echo $arrayInput[$y]['CC']; ?></td>
    <td align="right"><?php echo $arrayInput[$y]['FC']; ?></td>
    <td align="right"><?php echo $arrayInput[$y]['PC']; ?></td>
    <td align="right"><?php echo $arrayInput[$y]['BC']; ?></td>
  </tr>
  <?php
  } //end for
?>
  <tr>
    <td colspan="2"><b>Highest value : </b></td>
    <td align="right"><b><?php echo $maxCC; ?></b></td>
    <td align="right"><b><?php echo $maxFC; ?></b></td>
    <td align="right"><b><?php echo $maxPC; ?></b></td>
    <td align="right"><b><?php echo $maxBC; ?></b></td>
  </tr>

</table>

<?php
  $maxResultCC = max($resultCC);
  $maxResultFC = max($resultFC);
  $maxResultPC = max($resultPC);
  $maxResultBC = max($resultBC);

  if($totalRec == 0) {
    $insertValueMax =
    "INSERT INTO project_master.MAX_VALUE(
     DOC_ID, MAX_INPUT_CC, MAX_INPUT_FC, MAX_INPUT_PC, MAX_INPUT_BC,
     MAX_CRITERIA_VAL_CC, MAX_CRITERIA_VAL_FC, MAX_CRITERIA_VAL_PC,
     MAX_CRITERIA_VAL_BC
    ) VALUES (
      ".$maxDocId.",".$maxCC.", ".$maxFC.", ".$maxPC.", ".$maxBC.",
      ".$maxResultCC.", ".$maxResultFC.", ".$maxResultPC.", ".$maxResultBC."
    )";
    $insertValueMaxRs = mysql_query($insertValueMax, $con) or die(mysql_error());
  }
?>

<br/>
<center><input type="button" value="Generate Weightage" class="btn btn-large btn-
success" onclick="window.location='weightage.php?docId=<?php echo $maxDocId;
?>'"/></center>
<br/>

<?php

//close connection
mysql_close($con);
?>
  </body>
</html>
```

Appendix IV - d

weightage.php

```
<html lang="en">
<head>
  <meta charset="utf-8">
    <title>SSP Test Case Prioritization</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="">
    <meta name="author" content="">
    <!-- Le styles -->
    <link href="\bootstrap-3.3.5-dist\css\bootstrap.css" rel="stylesheet">
    <link href="\bootstrap-3.3.5-dist\css\justified-nav.css" rel="stylesheet">
    <link href="css/style2.css" rel="stylesheet" type="text/css" />
    <link href="css/style.css" rel="stylesheet" type="text/css" />
</head>
  <body>
    <div class="container">
      <div class="masthead">
        <h3 class="text-muted" align="center">Enhancing State-Sensitivity
        Partitioning by using Prioritization Approach</h3>
        <nav>
          <ul class="nav nav-justified">
                <li><a href="\tcpssp\home.php">Home</a></li>
                <li><a href="\tcpssp\form.php">Upload</a></li>
                <li><a href="\tcpssp\input.php">Output</a></li>
                <li  class="active"><a href="#">Criteria Value</a></li>
                <li><a href="#">Prioritized List</a></li>
                <li><a href="#">About</a></li>
          </ul>
        </nav>
      </div>

      <div class="jumbotron">
        <h2>Weightage and Test Cast Priority Value</h2>
        <p class="lead">Table 2 shows Coverage Criteria Value or Weightage given
        to every test sequence. The last column is Test Case Priority value
        calculated for every test sequence</p>
      </div>
<br/><br/>

<?php

include('conn.php');

$maxDocId = $_GET['docId'];

//select statement
$getPrioritizedList =
  "SELECT
    TC, TS, INPUT_CC, INPUT_FC, INPUT_PC, INPUT_BC, CRITERIA_VAL_CC,
    CRITERIA_VAL_FC, CRITERIA_VAL_PC, CRITERIA_VAL_BC, VAL_TCP, DATA_STATE
  FROM
    project_master.PRIORITIZED_LIST
  WHERE
    DOC_ID = '$maxDocId' ";
$getPrioritizedListRs = mysql_query($getPrioritizedList, $con) or
die(mysql_error());
```

```
?>
<br/>
<table class="table table-hover" border="1" cellspacing="0" cellpadding="3"
width="100%">
  <thead class="alert-info">
  <tr>
    <th rowspan="3">Test Case</th><th rowspan="3">Test Sequence</th>
    <th colspan="4">Test cases and its associated codes, functions, paths and
    branches</th>
    <th colspan="5">Test cases and its Criteria Value of codes, functions, paths
    and branches </th>
  </tr>
  <tr>
    <th colspan="4">Table 1</th>
    <th colspan="5">Table 2</th>
  </tr>
  <tr>
    <th>Code Coverage</th><th>Function coverage</th><th>Path
    Coverage</th><th>Branch Coverage</th>
    <th>Code Coverage Criteria Value</th><th>Function Coverage Criteria Value</th>
    <th>Path Coverage Criteria Value</th><th>Branch Coverage Criteria Value</th>
    <th>Test Case Priority Value </th>
  </tr>
  </thead>

  <?php
  while($dataArray = mysql_fetch_array($getPrioritizedListRs)) {
  $inputCC[] = $dataArray['INPUT_CC'];
  $inputFC[] = $dataArray['INPUT_FC'];
  $inputPC[] = $dataArray['INPUT_PC'];
  $inputBC[] = $dataArray['INPUT_BC'];

  $resultCC[] = $dataArray['CRITERIA_VAL_CC'];
  $resultFC[] = $dataArray['CRITERIA_VAL_FC'];
  $resultPC[] = $dataArray['CRITERIA_VAL_PC'];
  $resultBC[] = $dataArray['CRITERIA_VAL_BC'];

  if($dataArray['TC'] == '1') $style1 = "bg01";
  if($dataArray['TC'] == '2') $style1 = "bg02";
  if($dataArray['TC'] == '3') $style1 = "bg03";
  if($dataArray['TC'] == '4') $style1 = "bg04";
  if($dataArray['TC'] == '5') $style1 = "bg05";
  if($dataArray['TC'] == '6') $style1 = "bg06";
  if($dataArray['TC'] == '7') $style1 = "bg07";
  if($dataArray['TC'] == '8') $style1 = "bg08";
  if($dataArray['TC'] == '9') $style1 = "bg09";
  if($dataArray['TC'] == '10') $style1 = "bg10";
  ?>

  <tr class=<?php echo $style1; ?>>
    <td><?php echo $dataArray['TC']; ?></td>
    <td><?php echo $dataArray['TS']; ?></td>
    <td align="right"><?php echo $dataArray['INPUT_CC']; ?></td>
    <td align="right"><?php echo $dataArray['INPUT_FC']; ?></td>
    <td align="right"><?php echo $dataArray['INPUT_PC']; ?></td>
    <td align="right"><?php echo $dataArray['INPUT_BC']; ?></td>
    <td align="right"><?php echo $dataArray['CRITERIA_VAL_CC']; ?></td>
    <td align="right"><?php echo $dataArray['CRITERIA_VAL_FC']; ?></td>
```

```
        <td align="right"><?php echo $dataArray['CRITERIA_VAL_PC']; ?></td>
        <td align="right"><?php echo $dataArray['CRITERIA_VAL_BC']; ?></td>
        <td align="right"><?php echo $dataArray['VAL_TCP']." "; ?></td>
      </tr>
      <?php
        } //end while looping

      $maxCC = max($inputCC);
      $maxFC = max($inputFC);
      $maxPC = max($inputPC);
      $maxBC = max($inputBC);

      $maxResultCC = max($resultCC);
      $maxResultFC = max($resultFC);
      $maxResultPC = max($resultPC);
      $maxResultBC = max($resultBC);

      ?>
      <tr>
        <td colspan="2"><b>Highest value : </b></td>
        <td align="right"><b><?php echo $maxCC; ?></b></td>
        <td align="right"><b><?php echo $maxFC; ?></b></td>
        <td align="right"><b><?php echo $maxPC; ?></b></td>
        <td align="right"><b><?php echo $maxBC; ?></b></td>
        <td align="right"><b><?php echo $maxResultCC; ?></b></td>
        <td align="right"><b><?php echo $maxResultFC; ?></b></td>
        <td align="right"><b><?php echo $maxResultPC; ?></b></td>
        <td align="right"><b><?php echo $maxResultBC; ?></b></td>
        <td></td>
      </tr>
    </table>

    <br/>
    <center><input type="button" value="Generate Priority List" class="btn btn-large
    btn-success" onclick="window.location='prioritized.php?docId=<?php echo $maxDocId;
    ?>';"/></center>
    <br/>

    <?php
    //close connection
    mysql_close($con);
    ?>
      </body>
    </html>
```

Appendix IV - e

prioritized.php

```
<?php

include('conn.php');

$maxDocId = $_GET['docId'];

//select statement
$getPrioritizedList =
  "SELECT
    TC, TS, INPUT_CC, INPUT_FC, INPUT_PC, INPUT_BC,
    CRITERIA_VAL_CC, CRITERIA_VAL_FC, CRITERIA_VAL_PC, CRITERIA_VAL_BC, VAL_TCP,
    DATA_STATE
  FROM
    project_master.PRIORITIZED_LIST
  WHERE
    DOC_ID = '$maxDocId'
  ORDER BY
    VAL_TCP DESC, TC ASC ";

$getPrioritizedListRs = mysql_query($getPrioritizedList, $con) or
die(mysql_error());

?>
<html lang="en">
<head>
  <meta charset="utf-8">
    <title>SSP Test Case Prioritization</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="">
    <meta name="author" content="">

    <!-- Le styles -->
    <link href="\bootstrap-3.3.5-dist\css\bootstrap.css" rel="stylesheet">
    <link href="\bootstrap-3.3.5-dist\css\justified-nav.css" rel="stylesheet">
    <link href="css/style.css" rel="stylesheet" type="text/css" />
</head>

  <body>
    <div class="container">

      <div class="masthead">
        <h3 class="text-muted" align="center">Enhancing State-Sensitivity
        Partitioning by using Prioritization Approach</h3>
        <nav>
          <ul class="nav nav-justified">
                <li><a href="\tcpssp\home.php">Home</a></li>
                <li><a href="\tcpssp\input.php">Input</a></li>
                <li><a href="\tcpssp\output.php">Output</a></li>
                <li><a href="\tcpssp\weightage.php">Criteria Value</a></li>
                <li class="active"><a href="#">Prioritized List</a></li>
                <li><a href="#">About</a></li>
          </ul>
        </nav>
      </div>
      <div class="jumbotron">
        <h2>SSP : After Prioritization</h2>
        <p class="lead">List has been prioritized according to Test Case Priority
        Value.</p>

      </div>

<br/>
<table class="table table-hover" border="1" cellspacing="0" cellpadding="3"
width="100%">
<thead>
  <tr>
    <th rowspan="3">Test Case</th><th rowspan="3">Test Sequence</th>
    <th colspan="4">Test cases and its associated codes, functions, paths and
    branches</th>
    <th colspan="5">Test cases and its Criteria Value of codes, functions, paths
    and branches</th>
  </tr>
  <tr>
    <th colspan="4">Table 2</th>
    <th colspan="5">Table 3</th>
  </tr>
  <tr>
    <th>Code Coverage</th><th>Function coverage</th><th>Path
    Coverage</th><th>Branch Coverage</th>
    <th>Code Coverage Criteria Value</th><th>Function Coverage Criteria Value</th>
    <th>Path Coverage Criteria Value</th><th>Branch Coverage Criteria Value</th>
    <th>Test Case Priority Value </th>
  </tr>
  </thead>
  <?php
  while($dataArray = mysql_fetch_array($getPrioritizedListRs)) {
    $inputCC[] = $dataArray['INPUT_CC'];
    $inputFC[] = $dataArray['INPUT_FC'];
    $inputPC[] = $dataArray['INPUT_PC'];
    $inputBC[] = $dataArray['INPUT_BC'];

    $resultCC[] = $dataArray['CRITERIA_VAL_CC'];
    $resultFC[] = $dataArray['CRITERIA_VAL_FC'];
    $resultPC[] = $dataArray['CRITERIA_VAL_PC'];
    $resultBC[] = $dataArray['CRITERIA_VAL_BC'];

    if($dataArray['TC'] == '1') $style1 = "bg01";
    if($dataArray['TC'] == '2') $style1 = "bg02";
    if($dataArray['TC'] == '3') $style1 = "bg03";
    if($dataArray['TC'] == '4') $style1 = "bg04";
    if($dataArray['TC'] == '5') $style1 = "bg05";
    if($dataArray['TC'] == '6') $style1 = "bg06";
    if($dataArray['TC'] == '7') $style1 = "bg07";
    if($dataArray['TC'] == '8') $style1 = "bg08";
    if($dataArray['TC'] == '9') $style1 = "bg09";
    if($dataArray['TC'] == '10') $style1 = "bg10";
  ?>
  <tr class=<?php echo $style1; ?>>
    <td><?php echo $dataArray['TC']; ?></td>
    <td><?php echo $dataArray['TS']; ?></td>
    <td align="right"><?php echo $dataArray['INPUT_CC']; ?></td>
    <td align="right"><?php echo $dataArray['INPUT_FC']; ?></td>
    <td align="right"><?php echo $dataArray['INPUT_PC']; ?></td>
```

```
        <td align="right"><?php echo $dataArray['INPUT_BC']; ?></td>
        <td align="right"><?php echo $dataArray['CRITERIA_VAL_CC']; ?></td>
        <td align="right"><?php echo $dataArray['CRITERIA_VAL_FC']; ?></td>
        <td align="right"><?php echo $dataArray['CRITERIA_VAL_PC']; ?></td>
        <td align="right"><?php echo $dataArray['CRITERIA_VAL_BC']; ?></td>
        <td align="right"><?php echo $dataArray['VAL_TCP']." "; ?></td>

    </tr>
    <?php
      } //end while looping

      $maxCC = max($inputCC);
      $maxFC = max($inputFC);
      $maxPC = max($inputPC);
      $maxBC = max($inputBC);

      $maxResultCC = max($resultCC);
      $maxResultFC = max($resultFC);
      $maxResultPC = max($resultPC);
      $maxResultBC = max($resultBC);

    ?>
    <tr>
      <td colspan="2"><b>Highest value : </b></td>
      <td align="right"><b><?php echo $maxCC; ?></b></td>
      <td align="right"><b><?php echo $maxFC; ?></b></td>
      <td align="right"><b><?php echo $maxPC; ?></b></td>
      <td align="right"><b><?php echo $maxBC; ?></b></td>
      <td align="right"><b><?php echo $maxResultCC; ?></b></td>
      <td align="right"><b><?php echo $maxResultFC; ?></b></td>
      <td align="right"><b><?php echo $maxResultPC; ?></b></td>
      <td align="right"><b><?php echo $maxResultBC; ?></b></td>
      <td></td>
      <td></td>
    </tr>
</table>

<br/>
    <center>
      <input type="button" value="<<< Back" class="btn btn-large btn-success"
onclick="window.location='weightage.php?docId=<?php echo $maxDocId;
?>';"/>&nbsp;&nbsp;
      <input type="button" value="Export to CSV >>>" class="btn btn-large btn-
success" onclick="window.location='export.php?docId=<?php echo $maxDocId;
?>';"/></center>
<br/>
<?php
//close connection
mysql_close($con);
?>
    </body>
</html>
```

Appendix IV - f

output.php

```
<?php

include('conn.php');
include('func.php');

$maxDocId = $_GET['docId'];

//select statement
$getList =
  "SELECT
    TC, TS, INPUT_CC, INPUT_FC, INPUT_PC, INPUT_BC,
    CRITERIA_VAL_CC, CRITERIA_VAL_FC, CRITERIA_VAL_PC, CRITERIA_VAL_BC, VAL_TCP,
DATA_STATE
    FROM project_master.PRIORITIZED_LIST
    WHERE DOC_ID = '$maxDocId'
    ORDER BY VAL_TCP DESC, TC ASC";

$getListRs = mysql_query($getList, $con) or die(mysql_error());
$array_to_csv = array();

if(mysql_num_rows($getListRs) > 0) { //if it finds any row
  while($dataArray = mysql_fetch_array($getListRs)) {
  $header = array("TC","TS",
    "INPUT_CC","INPUT_FC","INPUT_PC","INPUT_BC",
    "CRITERIA_VAL_CC","CRITERIA_VAL_FC","CRITERIA_VAL_PC","CRITERIA_VAL_BC",
    "VAL_TCP","DATA_STATE");
  $row = array($dataArray['TC'],$dataArray['TS'],

$dataArray['INPUT_CC'],$dataArray['INPUT_FC'],$dataArray['INPUT_PC'],$dataArray['I
NPUT_BC'],$dataArray['CRITERIA_VAL_CC'],$dataArray['CRITERIA_VAL_FC'],$dataArray['
CRITERIA_VAL_PC'],$dataArray['CRITERIA_VAL_BC'],$dataArray['VAL_TCP'],$dataArray['
DATA_STATE']);

  $array_to_csv[] = array_combine($header,$row);

  //print_r($array_to_csv);
  }//end while looping
} //end mysql_num_rows

convert_to_csv($array_to_csv, 'output.csv', ',');

//close connection
mysql_close($con);

?>
```

func.php

```php
<?php

/**
 * Convert a comma separated file into an associated array.
 * The first row should contain the array keys. *
 * @param string $filename Path to the CSV file
 * @param string $delimiter The separator used in the file
 * @return array
 */

function csv_to_array($filename='', $delimiter=',')
{
  if(!file_exists($filename) || !is_readable($filename))
    return FALSE;
  $header = NULL;
  $data = array();
  if (($handle = fopen($filename, 'r')) !== FALSE)
  {
    while (($row = fgetcsv($handle, 1000, $delimiter)) !== FALSE)
    {
      if(!$header)
        $header = $row;
      else
        $data[] = array_combine($header, $row);
    }
    fclose($handle);
  }
  return $data;
}

/**
 * Convert an associated array into a comma separated file   *
 * @param array $input_array
 * @param string $output_file_name
 * @param string $delimiter The separator used in the file
 */
function convert_to_csv($input_array, $output_file_name, $delimiter)
{
    /** open raw memory as file, no need for temp files, be careful not to run out
    of memory thought */
    $f = fopen('php://memory', 'w');

    /** loop through array  */
    foreach ($input_array as $line) {
        /** default php csv handler **/
        fputcsv($f, $line, $delimiter);
    }
    /** rewrind the "file" with the csv lines **/
    fseek($f, 0);

    /** modify header to be downloadable csv file **/
    header('Content-Type: application/csv');
    header('Content-Disposition: attachement; filename="' . $output_file_name .
    '";');
```

```
        /** Send file to browser for download */
        fpassthru($f);
    }

           ?>
```