

|  |  |  |
| --- | --- | --- |
| MODULE NAME: |  | FUNDAMENTALS OF SOFTWARE DEVELOPMENT |
| MODULE CODE: |  | CT010-3-1-FSD |
| ASSIGNMENT TYPE: |  | GROUP ASSIGNMENT |
| INTAKE: |  | UC1F1611SE |
| LECTURER: |  | SUMAIRA MUHAMMAD HAYAT KHAN |
| HAND OUT DATE: |  | 21 DECEMBER 2016 |
| HAND IN DATE: |  | 20 MARCH 2017 |
| STUDENTS: |  | TP043972, MUSTAFA AHMED ABDULJABBAR  TP039934, RASHAD HELMI ABDULAZIZ AL-YOUSEFI  TP040205, MUKABAK ORAZBEK  TP040164, YOUSEF THARAWAT |

MARKING GRID

Group Work

|  |  |
| --- | --- |
| **Design**  **(15%)** |  |
| **Module Integration**  **(10%)** |  |
| **Documentation**  **(15%)** |  |
| **Total** |  |

Individual Work

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Student**  **Name & TP** | MUSTAFA TP043972 | RASHAD TP039934 | ORAZBEK TP040205 | YOUSEF TP040164 |
| **Program Listing**  **(15%)** |  |  |  |  |
| **Presentation**  **(15%)** |  |  |  |  |
| **Q & A**  **(15%)** |  |  |  |  |
| **Contribution**  **(15%)** |  |  |  |  |
| **Group marks** |  |  |  |  |
| **Total marks** |  |  |  |  |

WORK BREAKDOWN STRUCTURE

|  |  |  |
| --- | --- | --- |
| **Student**  **Name & TP** | **Responsible Parts** | **Signature** |
| MUSTAFA TP043972 | Order Meals. Report. File Sorter. Test Plan\*. |  |
| RASHAD TP039934 | Main Menu. Read Files. Conclusion. Test Plan\*. |  |
| ORAZBEK TP040205 | Introduction. Payment. Test Plan\*. |  |
| YOUSEF TP040164 | Order Page. Other Services. Test Plan\*. |  |

ABSTRACT

Group members have done this assignment to prepare a python program and to know more information, as well as to get knowledge regarding different codes of python. This report is a summary of what students have done in their whole project. Group have used the pseudocodes to explain every single part of project codes, furthermore, they have aided their writhing with a screenshots of the program. Also, members have did the test plan to check the outputs of every algorithms. Finally, students have commented to their coding parts to make it understandable.

**TABLE OF CONTENTS**

[MARKING GRID 2](#_Toc477776803)

[Group Work 2](#_Toc477776804)

[Individual Work 2](#_Toc477776805)

[WORK BREAKDOWN STRUCTURE 3](#_Toc477776806)

[ABSTRACT 4](#_Toc477776807)

[INTRODUCTION 6](#_Toc477776808)

[MAIN BODY 7](#_Toc477776809)

[Main Menu 7](#_Toc477776810)

[Display Main Menu 7](#_Toc477776811)

[Main Menu Commands 7](#_Toc477776812)

[Order Page 9](#_Toc477776813)

[Display Order Page 9](#_Toc477776814)

[Order Page Command Line 9](#_Toc477776815)

[Create Global Variables 11](#_Toc477776816)

[Read Files 12](#_Toc477776817)

[Get Items Name from List 13](#_Toc477776818)

[Get Items Price from List 14](#_Toc477776819)

[Sort Items Name by Alphabet 15](#_Toc477776820)

[Order Meals 16](#_Toc477776821)

[Display Meals 16](#_Toc477776822)

[Order Meals Basic Commands 17](#_Toc477776823)

[Order Meals 17](#_Toc477776824)

[Amounts of Meal 18](#_Toc477776825)

[Other Services 19](#_Toc477776826)

[Display Other Services 19](#_Toc477776827)

[Other Services Basic Commands 19](#_Toc477776828)

[Other Services Order 20](#_Toc477776829)

[Report 21](#_Toc477776830)

[Display Reports 21](#_Toc477776831)

[Report Basic Commands 21](#_Toc477776832)

[Payment 22](#_Toc477776833)

[Display Payment 22](#_Toc477776834)

[Payment Calculation 23](#_Toc477776835)

[Pay and Upload Data to Report 25](#_Toc477776836)

[test plan 26](#_Toc477776837)

[Introduction 26](#_Toc477776838)

[Team Members 26](#_Toc477776839)

[Test Environment 26](#_Toc477776840)

[Checking Process 26](#_Toc477776841)

[Testing Process 27](#_Toc477776842)

[Deliverables 28](#_Toc477776843)

[CONCLUSION 29](#_Toc477776844)

[REFFRENCES 30](#_Toc477776845)

INTRODUCTION

According to the researchers (Robertson, 2007), computer programming is an art. Unfortunately, many people believe that a programmer must be good in calculations and must figure out more about technical information, as well as must be prepared to spend long hours in front of computer screen. However, given the right tools and steps to follow, anyone can write well designed programs. It is a task worth doing, as it is both stimulating and fulfilling. In order to create a valuable software, developers should use their intellectual and imaginational skills, and it is the main philosophy of success in any case.

MAIN BODY

Main Menu

Display Main Menu

**Pseudocode:**

PROGRAM Main\_Menu\_Display:

DISPLAY "MAIN MENU";

DISPLAY "(O) ORDER";

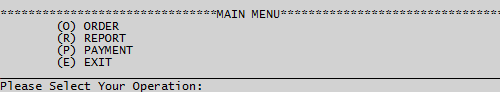
DISPLAY "(R) REPORT";

DISPLAY "(P) PAYMENT";

DISPLAY "(E) EXIT";

END.

**Outputs:**



*Main Menu*

Main Menu Commands

**Pseudocode:**

PROGRAM Main\_Menu\_Command:

DOWHILE TRUE:

Var\_Input = STRING(INPUT("Please Select Your Operation: ")) UPPER;

IF LENGTH(Var\_Input) == 1 THEN:

IF Var\_Input == "O" THEN:

ORDER\_MENU;

BREAK;

ELSEIF Var\_Input == "R" THEN:

REPORT;

BREAK;

ELSEIF Var\_Input == "P" THEN:

PAYMENT;

BREAK;

ELSEIF Var\_Input == "E" THEN:

DISPLAY "THANK YOU";

BREAK;

ELSE:

DISPLAY "ERROR: Invalid Input (" + STRING(Var\_Input) + "). TRY again!";

ENDIF.

ELSE:

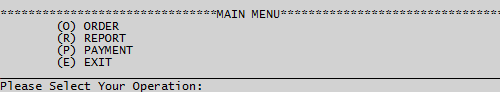
DISPLAY "ERROR: Invalid Input (" + STRING(Var\_Input) + "). TRY again!";

ENDIF.

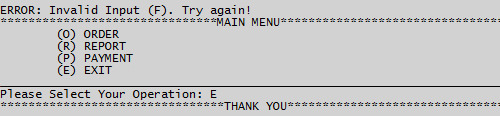
ENDWHILE.

END.

**Outputs:**



*Main Menu with Command Line*



*Exit form Main Menu*

Order Page

Display Order Page

**Pseudocode:**

PROGRAM Order\_Menu\_Display:

DISPLAY "ORDER PAGE"

DISPLAY "(F) FOODS AND DRINKS"

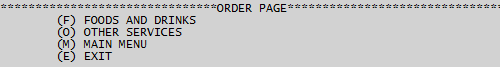
DISPLAY "(O) OTHER SERVICES"

DISPLAY "(M) MAIN MENU"

DISPLAY "(E) EXIT"

END.

**Outputs:**



*Order Page*

Order Page Command Line

**Pseudocode:**

PROGRAM Order\_Menu\_Display:

DOWHILE TRUE:

Var\_Input = STRING(INPUT("Please Select Your Operation: ")) UPPER;

IF LENGTH(Var\_Input) == 1 THEN:

IF Var\_Input == "F" THEN:

ORDER\_MEALS;

BREAK;

ELSEIF Var\_Input == "O" THEN:

OTHER\_SERVICES;

BREAK;

ELSEIF Var\_Input == "M" THEN:

MAIN\_MENU;

BREAK;

ELSEIF Var\_Input == "E" THEN:

DISPLAY "THANK YOU";

BREAK;

ELSE:

DISPLAY "ERROR: Invalid Input (" + STRING(Var\_Input) + "). TRY again!";

ENDIF.

ELSE:

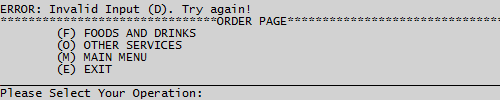
DISPLAY "ERROR: Invalid Input (" + STRING(Var\_Input) + "). TRY again!";

ENDIF.

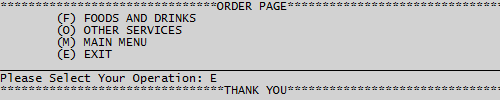
ENDWHILE.

END.

**Outputs:**

****

*Order Page Command Line*

**

*Exit from Order Page*

Create Global Variables

**Pseudocode:**

PROGRAM Create\_Global\_Variables:

GLOBAL List\_Drinks;

GLOBAL List\_Foods;

GLOBAL List\_Services;

GLOBAL List\_Item\_Order;

GLOBAL List\_Item\_Price;

END.

Read Files

**Pseudocode:**

PROGRAM File\_Reader:

GLOBAL List\_Foods;

GLOBAL List\_Drinks;

GLOBAL List\_Services;

File\_Foods = OPEN("FOODS.FSD", "READ MODE");

FOR i IN File\_Foods THEN:

List\_Foods.APPEND(STRING(i.STRIP));

ENDFOR.

File\_Foods.CLOSE;

File\_Drinks = OPEN("DRINKS.FSD", "READ MODE");

FOR i IN File\_Drinks THEN:

List\_Drinks.APPEND(STRING(i.STRIP));

ENDFOR.

File\_Drinks.CLOSE;

File\_Services = OPEN("SERVICES.FSD", "READ MODE");

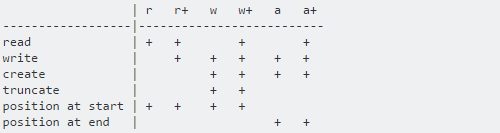
FOR i IN File\_Services THEN:

List\_Services.APPEND(STRING(i.STRIP));

ENDFOR.

File\_Services.CLOSE;

END.



*Six different Modes*

Get Items Name from List

**Pseudocode:**

PROGRAM Get\_Items\_Name:

GLOBAL List\_Foods;

GLOBAL List\_Drinks;

GLOBAL List\_Services;

List\_Foods = ["Foods\_Name\_RM\_00.00", "", "", ...];

List\_Drinks = ["Drinks\_Name\_RM\_00.00", "", "", ...];

List\_Services = ["Services\_Name\_RM\_00.00", "", "", ...];

i = 0;

DOWHILE i <= LENGTH(List\_Foods) - 1:

IF "RM" IN List\_Foods[i] THEN:

List\_Foods[i] = STRING(List\_Foods[i][:List\_Foods[i].INDEX("RM") - 1]);

ENDIF.

STEP 1;

ENDWHILE.

i = 0;

DOWHILE i <= LENGTH(List\_Drinks) - 1:

IF "RM" IN List\_Drinks[i] THEN:

List\_Drinks[i] = STRING(List\_Drinks[i][:List\_Drinks[i].INDEX("RM") - 1]);

ENDIF.

STEP 1;

ENDWHILE.

i = 0;

DOWHILE i <= LENGTH(List\_Services) - 1:

IF "RM" IN List\_Services[i] THEN:

List\_Services[i] = STRING(List\_Services[i][:List\_Services[i].INDEX("RM") - 1]);

ENDIF.

STEP 1;

ENDWHILE.

END.

Get Items Price from List

**Pseudocode:**

PROGRAM Get\_Items\_Price:

GLOBAL List\_Foods;

GLOBAL List\_Drinks;

GLOBAL List\_Services;

List\_Foods = ["Foods\_Name\_RM\_00.00", "", "", ...];

List\_Drinks = ["Drinks\_Name\_RM\_00.00", "", "", ...];

List\_Services = ["Services\_Name\_RM\_00.00", "", "", ...];

i = 0;

DOWHILE i < LENGTH(List\_Foods):

List\_Item\_Price[i] = List\_Foods[i][INTEGER(List\_Foods[i].INDEX("RM") + 3):];

STEP 1;

ENDWHILE.

i = 0;

DOWHILE i < LENGTH(List\_Drinks):

List\_Item\_Price[40 + i] =

List\_Drinks[i][INTEGER(List\_Drinks[i].INDEX("RM") + 3):];

STEP 1;

ENDWHILE.

i = 0;

DOWHILE i < LENGTH(List\_Services):

List\_Item\_Price[80 + i] =

List\_Services[i][INTEGER(List\_Services[i].INDEX("RM") + 3):];

STEP 1;

ENDWHILE.

END.

Sort Items Name by Alphabet

**Pseudocode:**

PROGRAM Items\_Name\_Sorter:

GLOBAL List\_Foods;

GLOBAL List\_Drinks;

GLOBAL List\_Services;

List\_Foods = ["Foods\_Name\_RM\_00.00", "", "", ...];

List\_Drinks = ["Drinks\_Name\_RM\_00.00", "", "", ...];

List\_Services = ["Services\_Name\_RM\_00.00", "", "", ...];

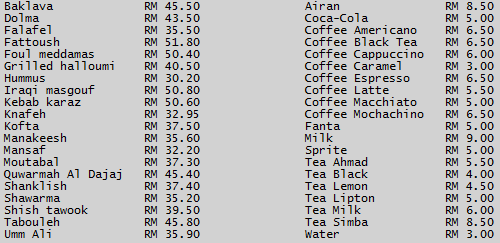
List\_Foods = SORTED(List\_Foods);

List\_Drinks = SORTED(List\_Drinks);

List\_Services = SORTED(List\_Services);

END.

**Outputs:**



*Sorted Item Names*

Order Meals

Display Meals

**Pseudocode:**

PROGRAM Order\_Meals\_Display:

List\_Foods = ["Foods\_Name\_RM\_00.00", "", "", ...];

List\_Drinks = ["Drinks\_Name\_RM\_00.00", "", "", ...];

DOWHILE TRUE:

DISPLAY "ORDER FOODS & DRINKS";

DISPLAY " |NO| |FOOD NAME| |PRICE| | |NO| |DRINK NAME| |PRICE|";

i = 0;

DOWHILE i < LENGTH(List\_Foods) OR i < LENGTH(List\_Drinks):

IF i < LENGTH(List\_Foods) THEN:

Food = STRING(i + 1) + STRING(List\_Foods[i]);

ELSE:

Food = " " \* 36;

ENDIF.

IF i < LENGTH(List\_Drinks) THEN:

Drink = STRING(41 + i) + STRING(List\_Drinks[i]);

ELSE:

Drink = "";

ENDIF.

DISPLAY Food, Drink;

STEP 1;

DISPLAY "(M) MAIN MENU";

DISPLAY "(P) PAYMENT";

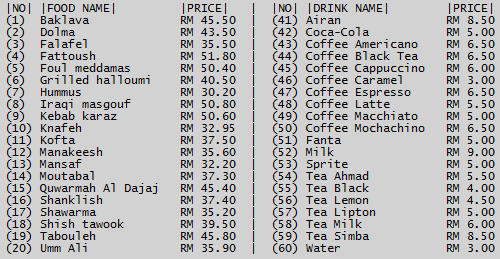
DISPLAY "(E) EXIT";

ENDWHILE.

ENDWHILE.

END.

**Outputs:**



*Meals List*

Order Meals Basic Commands

**Pseudocode:**

PROGRAM Order\_Meals\_Basic\_Command:

DOWHILE TRUE:

Var\_Input = INPUT("Please Select Your Operation: ") UPPER;

IF Var\_Input == "M" THEN:

MAIN\_MENU;

BREAK;

ENDIF.

IF Var\_Input == "E" THEN:

DISPLAY "THANK YOU";

BREAK;

ENDIF.

IF Var\_Input == "P" THEN:

PAYMENT;

BREAK;

ENDIF.

ENDWHILE.

END.

**Outputs:**



*Basic Commands*

Order Meals

**Pseudocode:**

PROGRAM Order\_Meals\_Order\_Command:

List\_Foods = ["Foods\_Name\_RM\_00.00", "", "", ...];

List\_Drinks = ["Drinks\_Name\_RM\_00.00", "", "", ...];

DOWHILE TRUE:

INTEGER(Var\_Input);

IF (INTEGER(Var\_Input) <= LENGTH(List\_Foods) AND INTEGER(Var\_Input) > 0)

OR (INTEGER(Var\_Input) <= LENGTH(List\_Drinks) + 40

AND INTEGER(Var\_Input) > 40) THEN:

TRY:

DISPLAY STRING(List\_Foods[INTEGER(Var\_Input) - 1]);

EXCEPT:

PASS;

TRY:

DISPLAY STRING(List\_Drinks[INTEGER(Var\_Input) - 41]);

EXCEPT:

PASS;

ELSE:

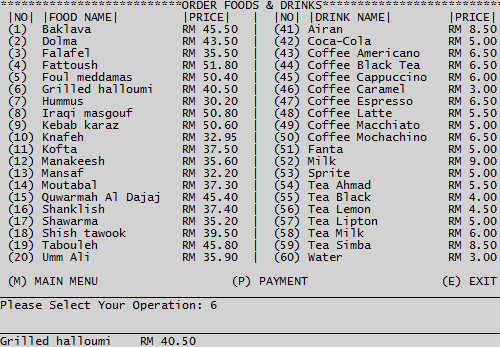
DISPLAY "ERROR: Invalid Input (" + STRING(Var\_Input) + "). TRY again!";

ENDIF.

ENDWHILE.

END.

**Outputs:**



*Choosing Meals*

Amounts of Meal

**Pseudocode:**

PROGRAM Order\_Meals\_Amount\_Command:

DOWHILE TRUE:

Var\_Input\_1 = INPUT("How Many You Want to Order?: ");

IF INTEGER(Var\_Input\_1) > 0 THEN:

List\_Item\_Order[INTEGER(Var\_Input) - 1] += INTEGER(Var\_Input\_1);

DISPLAY "Successfully Ordered!";

ORDER\_MEALS;

BREAK;

ELSE:

DISPLAY "ERROR: Invalid Input (" + STRING(Var\_Input\_1) + "). TRY again!";

ENDIF.

ENDWHILE.

END.

**Outputs:**



*Amounts of Meal*

Other Services

Display Other Services

**Pseudocode:**

PROGRAM Other\_Services\_Display:

List\_Services = ["Services\_Name\_RM\_00.00", "", "", ...];

DOWHILE TRUE:

DISPLAY "OTHER SERVICES";

DISPLAY " |NO| |SERVICE NAME| |PRICE|";

i = 0;

DOWHILE i < LENGTH(List\_Services):

DISPLAY STRING(81 + i) + STRING(List\_Services[i]);

STEP 1;

ENDWHILE.

DISPLAY "(M) MAIN MENU";

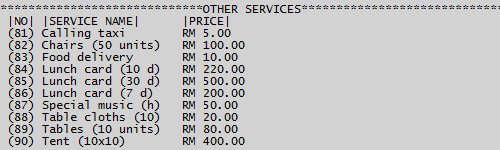
DISPLAY "(P) PAYMENT";

DISPLAY "(E) EXIT";

ENDWHILE.

END.

**Outputs:**



*Other Services*

Other Services Basic Commands

**Pseudocode:**

PROGRAM Other\_Services\_Basic\_Command:

DOWHILE TRUE:

Var\_Input = INPUT("Please Select Your Operation: ") UPPER;

IF Var\_Input == "M" THEN:

MAIN\_MENU;

BREAK;

ENDIF.

IF Var\_Input == "E" THEN:

DISPLAY "THANK YOU";

BREAK;

ENDIF.

IF Var\_Input == "O" THEN:

PAYMENT;

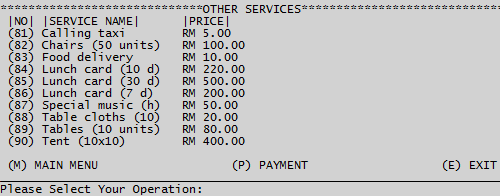
BREAK;

ENDIF.

ENDWHILE.

END.

**Outputs:**



*Other Services Commands*

Other Services Order

**Pseudocode:**

PROGRAM Other\_Services\_Order\_Command:

List\_Services = ["Services\_Name\_RM\_00.00", "", "", ...];

DOWHILE TRUE:

TRY:

INTEGER(Var\_Input);

IF INTEGER(Var\_Input) > 80 AND INTEGER(Var\_Input) < 100 THEN:

DISPLAY "Successfully Ordered: ";

DISPLAY STRING(List\_Services[INTEGER(Var\_Input) - 81]);

List\_Item\_Order[INTEGER(Var\_Input) - 1] = 1;

OTHER\_SERVICES;

BREAK;

ELSE:

DISPLAY "ERROR: Invalid Input (" + STRING(Var\_Input) + "). TRY again!";

ENDIF.

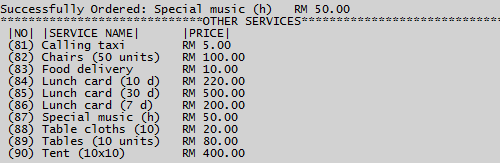
EXCEPT:

DISPLAY "ERROR: Invalid Input (" + STRING(Var\_Input) + "). TRY again!";

ENDWHILE.

END.

**Outputs:**



*Other Services Ordered*

Report

Display Reports

**Pseudocode:**

PROGRAM Report\_Display

DOWHILE TRUE:

DISPLAY "REPORT";

File\_Report = OPEN("REPORT.FSD", "READ MODE").READ;

DISPLAY File\_Report;

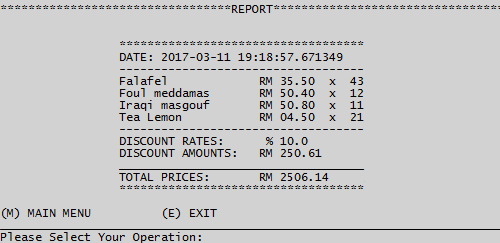
DISPLAY "(M) MAIN MENU";

DISPLAY "(E) EXIT";

ENDWHILE.

END.

**Outputs:**



*Reports*

Report Basic Commands

**Pseudocode:**

PROGRAM Report\_Basic\_Command:

DOWHILE TRUE:

Var\_Input = STRING(INPUT("Please Select Your Operation: "));

IF "M" IN Var\_Input OR "m" IN Var\_Input THEN:

MAIN\_MENU;

BREAK;

ELSEIF "E" IN Var\_Input OR "e" IN Var\_Input THEN:

DISPLAY "THANK YOU";

BREAK;

ELSE:

DISPLAY "ERROR: Invalid Input (" + STRING(Var\_Input) + "). TRY again!";

ENDIF.

ENDWHILE.

END.

Payment

Display Payment

**Pseudocode:**

PROGRAM Payment\_Display:

DOWHILE TRUE:

DISPLAY "PAYMENT";

Total\_Price = 0;

Report\_New = "DATE:" + STRING(TIME.NOW);

i = 0;

DOWHILE i < LENGTH(List\_Item\_Order):

IF List\_Item\_Order[i] != 0 THEN:

IF i >= 0 AND i < 40 THEN:

Report\_New += STRING(List\_Foods[i]) + STRING(List\_Item\_Order[i]);

DISPLAY STRING(List\_Foods[i]) + STRING(List\_Item\_Order[i]);

Total\_Price += List\_Item\_Price[i] \* List\_Item\_Order[i];

ENDIF.

IF i >= 40 AND i < 80 THEN:

Report\_New += STRING(List\_Drinks[i - 40]) + STRING(List\_Item\_Order[i]);

DISPLAY STRING(List\_Drinks[i - 40]) + STRING(List\_Item\_Order[i]);

Total\_Price += List\_Item\_Price[i] \* List\_Item\_Order[i];

ENDIF.

IF i >= 80 AND i < 100 THEN:

Report\_New += STRING(List\_Services[i - 80]);

DISPLAY STRING(List\_Services[i - 80]);

Total\_Price += List\_Item\_Price[i] \* List\_Item\_Order[i];

ENDIF.

STEP 1;

ELSE:

STEP 1;

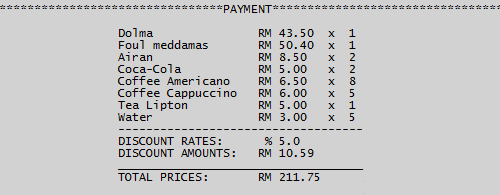
ENDIF.

ENDWHILE.

ENDWHILE.

END.

**Outputs:**



*Bill*

Payment Calculation

**Pseudocode:**

PROGRAM Payment\_Calculation\_With\_Discounts:

Var\_Discount\_1 = 200;

Var\_Discount\_2 = 1000;

Var\_Discount\_3 = 5000;

Var\_Discount\_1\_Rate = 5 %;

Var\_Discount\_2\_Rate = 10 %;

Var\_Discount\_3\_Rate = 15 %;

DOWHILE TRUE:

IF Total\_Price > Var\_Discount\_3 THEN:

Total\_Price -= Total\_Price \* Var\_Discount\_3\_Rate;

Report\_New += "DISCOUNT RATES: %" + STRING(Var\_Discount\_3\_Rate \* 100);

Report\_New += "DISCOUNT AMOUNTS: RM" +

STRING(ROUND(Total\_Price \* Var\_Discount\_3\_Rate, 2));

Report\_New += "TOTAL PRICES: RM" + STRING(ROUND(Total\_Price, 2)

Report\_New += "DISCOUNT RATES: %" + STRING(Var\_Discount\_3\_Rate \* 100);

Report\_New += "DISCOUNT AMOUNTS: RM" +

STRING(ROUND(Total\_Price \* Var\_Discount\_3\_Rate, 2));

Report\_New += "TOTAL PRICES: RM" + STRING(ROUND(Total\_Price, 2)));

ELSEIF Total\_Price > Var\_Discount\_2 THEN:

Total\_Price -= Total\_Price \* Var\_Discount\_2\_Rate;

Report\_New += "DISCOUNT RATES: %" + STRING(Var\_Discount\_2\_Rate \* 100);

Report\_New += "DISCOUNT AMOUNTS: RM" +

STRING(ROUND(Total\_Price \* Var\_Discount\_2\_Rate, 2));

Report\_New += "TOTAL PRICES: RM" + STRING(ROUND(Total\_Price, 2);

Report\_New += "DISCOUNT RATES: %" + STRING(Var\_Discount\_2\_Rate \* 100);

Report\_New += "DISCOUNT AMOUNTS: RM" +

STRING(ROUND(Total\_Price \* Var\_Discount\_2\_Rate, 2));

Report\_New += "TOTAL PRICES: RM" + STRING(ROUND(Total\_Price, 2)));

ELSEIF Total\_Price > Var\_Discount\_1 THEN:

Total\_Price -= Total\_Price \* Var\_Discount\_1\_Rate;

Report\_New += "DISCOUNT RATES: %" + STRING(Var\_Discount\_1\_Rate \* 100);

Report\_New += "DISCOUNT AMOUNTS: RM" +

STRING(ROUND(Total\_Price \* Var\_Discount\_1\_Rate, 2));

Report\_New += "TOTAL PRICES: RM" + STRING(ROUND(Total\_Price, 2)

Report\_New += "DISCOUNT RATES: %" + STRING(Var\_Discount\_1\_Rate \* 100);

Report\_New += "DISCOUNT AMOUNTS: RM" +

STRING(ROUND(Total\_Price \* Var\_Discount\_1\_Rate, 2));

Report\_New += "TOTAL PRICES: RM" + STRING(ROUND(Total\_Price, 2)));

ELSE:

Report\_New += "TOTAL PRICES: RM" + STRING(ROUND(Total\_Price, 2));

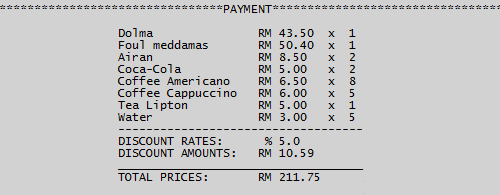
DISPLAY "TOTAL PRICES: RM" + STRING(ROUND(Total\_Price, 2));

ENDIF.

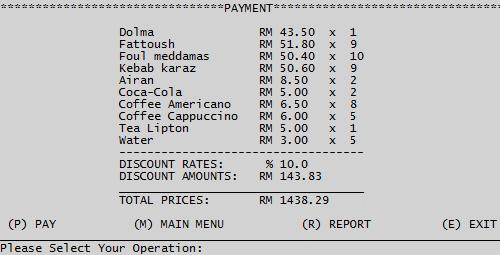
ENDWHILE.

END.

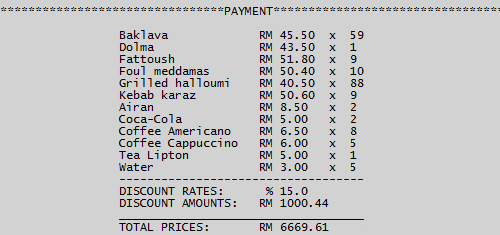
**Outputs:**



*Bill with 5% Discount (after RM 200)*



*Bill with 10% Discount (after RM 1000)*



*Bill with 15% Discount (after RM 5000)*

Pay and Upload Data to Report

**Pseudocode:**

PROGRAM Upload\_Data:

Var\_Discount\_1 = 200;

Var\_Discount\_2 = 1000;

Var\_Discount\_3 = 5000;

Var\_Discount\_1\_Rate = 5 %;

Var\_Discount\_2\_Rate = 10 %;

Var\_Discount\_3\_Rate = 15 %;

DOWHILE TRUE:

DISPLAY "(P) PAY";

DISPLAY "(M) MAIN MENU";

DISPLAY "(R) REPORT";

DISPLAY "(E) EXIT";

Var\_Input = STRING(INPUT("Please Select Your Operation: ")) UPPER;

IF Var\_Input == "P" THEN

DISPLAY "Successfully Paid!";

File\_Report = OPEN("REPORT.FSD", "WRITE MODE");

File\_Report.WRITE(Report\_New);

File\_Report.CLOSE;

ELSEIF Var\_Input == "M" THEN

MAIN\_MENU;

BREAK;

ELSEIF Var\_Input == "R" THEN

REPORT;

BREAK;

ELSEIF Var\_Input == "E" THEN

DISPLAY "THANK YOU";

BREAK;

ELSE:

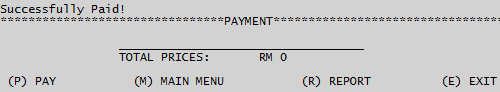
DISPLAY "ERROR: Invalid Input (" + STRING(Var\_Input) + "). TRY again!";

ENDIF.

ENDWHILE.

END.

**Outputs:**



*Successfully Paid!*

test plan

**Project:** Restaurant Controller

**Date:** 14/3/2017

Introduction

The Test Plan has been created to communicate the test approach to team members. It includes team members list and role of them, test environment, test process and deliverables. This document will clearly identify the difference between expected results and actual results.

Team Members

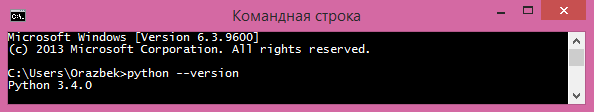
|  |  |
| --- | --- |
| **MEMBERS** | **ROLE** |
| MUSTAFA AHMED ABDULJABBAR | Developer, Data Base Manager |
| RASHAD HELMI ABDULAZIZ AL-YOUSEFI | Developer, User Interface Designer |
| MUKABAK ORAZBEK | Developer, Project Manager |
| YOUSEF THARAWAT | Developer, Tester |

Test Environment

* A computer required.
* Empty space with 9 MB (1 MB for software, 8 MB for user data).
* 10 MB Memory.
* Python 2.7 or higher versions are required.

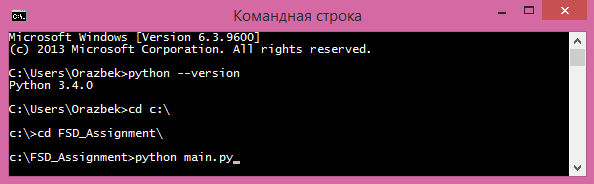
Checking Process

Checking python version:



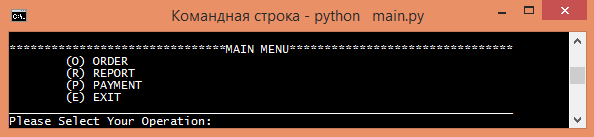
*Python Version*

Open project:



*Open project*

Running project:



*Running project*

Testing Process

|  |  |  |
| --- | --- | --- |
| **ACTIONS** | **EXPECTED RESULTS** | **ACTUAL RESULTS** |
| **Input Test**  In this action, testers tried to insert risky symbols or key words to input operation.  Input symbol is: = | The algorithm need to identify invalid inputs, and display it on the screen. |  |
| **Sorting Items Name**  In this action, testers tried to test sorting function, and replaced all items name randomly. | Expected results is algorithm need to arrange all items name by alphabetic order. |  |
| **Display Ordered Items**  In this step, testers tried to check calculation part of payment. Firstly need to order some items. More than 1. | Expected result is:  45.5 \* 2 = 91  35.5 \* 2 = 71  6 \* 3 = 18  Total: 180 |  |
| **Upload Data**  In this action, testers tried to upload data to into report file. | Expected results are:  Data have to locate into report.fsd file with uploading time, and have to show all order details |  |

Deliverables

|  |  |
| --- | --- |
| **DELIVERABLE** | **FOR** |
| Test Plan | Tester Team |
| Test Status report | Project Manager, Developer |
| Test Results | All team members |

CONCLUSION

In this documentation, students have prepared a python coding program. They have used a file which will enable to users to add or delete anything from the items menu. Furthermore, developers have used global variables, statements, conditions to minimize algorithms, and running it smoothly. Also, programmers used different functions for each actions, such as order, payment or report. Finally, students have tested their whole project to find differences between expected results and actual results, then used different risky inputs and testing environments to make sure the quality of algorithms.

REFFRENCES

Robertson, L.A. (2007) *Simple Program Design: A Step-by-Step Approach.* Fifth edition. Thomson. Nelson Australia Pty Limited.

Thenewboston. (2009) *Python Programming Tutorial: Video 1 – 43*. [Online] Available at: https://thenewboston.com/videos.php?cat=36.

[Accessed: 01/03/2017].

Python 3.4.5 Documentation. (2016). *Standard Library OS: Miscellaneous Operating System Interfaces.* [Online] Available at: https://docs.python.org/3.4/library/os.html.

[Accessed: 24/02/2017].

Stackoverflow. (2010) *Python Open Built-in Function: Difference between Modes a, a+, w, w+, and r+?* [Online] Available at: http://stackoverflow.com/questions/1466000/python-open-built-in-function-difference-between-modes-a-a-w-w-and-r.

[Accessed: 21/02/2017].