Department of Software Technology

CCPROG1 Machine Specifications

Deadline: December 5, 2022 (Monday) 8:00AM via AnimoSpace

Restaurants and fast-food chains have been using ordering systems for handling orders. These kinds of systems not only takes inputs of the orders but also computes the amount due for the order. This project will have you develop such a system.

The following tables will show the food items available and their prices.

Mains:

ID	Type	Price
1	Chicken	90.00
2	Pork	105.00
3	Fish	120.00
4	Beef	135.00

Table 1: List of main dishes and their corresponding prices

Sides:

ID	Type	Price
1	Steamed Rice	20.00
2	Shredded Corn	35.00
3	Mashed Potatoes	50.00
4	Steam Vegetables	65.00

Table 2: List of side dishes and their corresponding prices

Drinks:

ID	Type	Price
1	Mineral Water	25.00
2	Iced Tea	35.00
3	Soda	45.00
4	Fruit Juice	55.00

Table 3: List of drink and their corresponding prices

Order – An order is the process where the user will input 1 main, 1 side and 1 drink.

Meal Set of the day – A meal set that is randomly selected by the system for the day. Customers who order the set of the day will have a discount of 15% off for that specific set. This set will change daily.

Features:

Ordering:

A customer can make three(3) orders or less. They are allowed to only order specific items from either the main, sides or drinks(eg. Only order a main and nothing else).

Example:

```
Order 1:
      Main:      3
            Fish
      Side:      0
            None
      Drink:     0
            None

Order 2:
      Main:      0
            None
      Side:      3
            Mashed Potatoes
      Drink:     4
            Fruit Juice

Order 3:
      Main:      1
            Chicken
      Side:      1
            Steamed Rice
      Drink:     0
            None
```

Each type is limited to 1 piece. (eg. 2 mains in 1 order is not allowed).

Cancelling:

The system will allow the user to cancel an order.

- The system will have the option to let the user cancel an order at the start of the next order. The system will then proceed to the Amount Due part.

  Example: The customer initially decided to make 2 orders but then decides that he/she only wants to make order 1 instead. After selecting the items for the first order(main, side and drink), the user may select the cancel option for the second order.

- The system will have also have the option to let the user cancel an order after all orders have finished OR before the start of the first order. The system will then go back to the Ordering part.

Modifying/Changing Order of a Meal Set:

The system will allow a specific order to be modified. This will happen only after finishing inputting each item for that order. Specifically, after the input for the drink. The system will prompt the user to confirm if the order is correct. If the user selects no, then the system will repeat the process for that order.

Example:

```
Order 1:
     Main:      1
          Pork
     Side:      2
          Shredded Corn
     Drink:     3
          Soda

Is this meal set order correct(Y/N)? N

Order 1:
     Main:      3
          Pork
     Side:      4
          Steamed Vegetables
     Drink:     1
          Mineral Water

Is this meal set order correct(Y/N)? Y

Order 2:
     Main:…
```

Total Amount Due:

The system will be able to display the subtotals and total amount that the customer must pay for their order. If a customer orders the Meal Set of the day, the discount will be listed under that set.

Example:

```
Order 1:
      Main:      Fish             P120.00
      Side:      Shredded Corn    P35.00
      Drink:     Soda             P45.00
      Meal Set of the day Discount 15%
Subtotal:                                    P170.00

Order 2:
      Main:      Pork             P105.00
      Side:      Mashed Potatos   P50.00
      Drink:     None             P0.00
Subtotal:                                    P155.00


Total Amount Due:                            P325.00
```

How to Approach the Machine Project

Step 1: Problem analysis and algorithm formulation

Read the MP Specifications again! Identify clearly what are the required information from the user, what kind of processes are needed, and what will be the output (s) of your program. Clarify with your professor any issues that you might have regarding the machine project.

When you have all the necessary information, identify the necessary functions that you will need to modularize the project. Identify the required data of these functions and what kind of data they will return to the caller. Write your algorithm for each of these modules/functions as well as the algorithm for your main program.

Step 2: Implementation

In this step, you are to translate your algorithm into proper C statements. While implementing, you are to perform the other phases of program planning and design (discussed in the other steps below) together with this step.

Follow the coding standard indicated in the course notes (Modules section in AnimoSpace).

You may choose to type your program in a text editor or an IDE (i.e. Dev-C IDE) at this point. Note that you are expected to use statements taught in class. You can explore other libraries and functions in C as long as you can clearly explain how these work. You may also use arrays, should these be applicable and

you are able to properly justify and explain your implementation using these. For topics not covered, it is left to the student to read ahead, research, and explore by himself.

Note though that you are NOT ALLOWED to do the following:

- to declare and use global variables (i.e., variables declared outside any function),
- to use goto statements (i.e., to jump from code segments to code segments),
- to use the break or continue statement to exit a block. Break statement can only be used to break away from the switch block,
- to use the return statement or exit statement to prematurely terminate a loop or function or program,
- to use the exit statement to prematurely terminate a loop or to terminate the function or program, and
- to call the main() function to repeat the process instead of using loops.

It is best that you perform your coding "incrementally." This means:

- Dividing the program specification into subproblems, and solving each problem separately according to your algorithm;
- Code the solutions to the subproblems one at a time. Once you're done coding the solution for one subproblem, apply testing and debugging.

Documentation

While coding, you have to include internal documentation in your programs. You are expected to have the following:

- File comments or Introductory comments
- Function comments
- In-line comments

Introductory comments are found at the very beginning of your program before the preprocessor directives. Follow the format shown below. Note that items in between < > should be replaced with the proper information.

```
/*
     Description:      <Describe what this program does briefly>
     Programmed by:    <your name here>   <section>
     Last modified:    <date when last revision was made>
     Version:          <version number>
    [Acknowledgements: <list of sites or borrowed libraries and sources>]
*/
<Preprocessor directives>

<function implementation>

int main()
{
   return 0;
}
```

Function comments precede the function header. These are used to describe what the function does and the intentions of each parameter and what is being returned, if any. If applicable, include

pre-conditions as well. Pre-conditions refer to the assumed state of the parameters. Follow the format below when writing function comments:

```
/*    <Description of function>
      Precondition: <precondition / assumption>
      @param <name> <purpose>
      @return <description of returned result>
*/
<return type>
<function name> (<parameter list>)
      :
```

Example:

```
/* This function computes for the area of a triangle
   Precondition: base and height are non-negative values
   @param base is the base measurement of the triangle in cm
   @param height is the height measurement of the triangle in
cm
   @return the resulting area of the triangle
*/
float
getAreaTri (float base,
            float height)
{
    ...
}
```

In-Line Comments are ot**her comments in major parts of the code. These are expected to explain the purpose or algorithm of groups of related code, esp. for long functions.**

Step 3: Testing and Debugging

Submit the list of test cases you have used. For each feature of your program, you have to fully test it before moving to the next feature. Sample questions that you should ask yourself are:

1. What should be displayed on the screen if the user inputs an order?
2. What would happen if I input incorrect inputs? (e.g., values not within the range)
3. Is my program displaying the correct output?
4. Is my program following the correct sequence of events (correct program flow)?
5. Is my program terminating (ending/exiting) correctly? Does it exit when I press the command to `quit`? Does it exit when the program's goal has been met? Is there an infinite loop?
6. and others…

Important Points to Remember:

1. You are required to implement the project using the C language (C99 and NOT C++). Make sure you know how to compile and run in both the IDE (DEV-C++) and the command prompt (via
   `gcc –Wall <yourMP.c> -o <yourExe.exe>`
2. The implementation will require you to:
   - Create and Use Functions
     Note: Non-use of self-defined functions will merit a grade of **0** for the **machine project**. Too few self-defined functions may merit deductions. A general rule is to create a separate function for each option described above, unless some features are too similar that one function can serve the purpose for two [or more] of the options. Note that functions whose tasks are only to display are not included in the count for creating user-defined functions.
   - Appropriately use conditional statements, loops and other constructs discussed in class (Do not use brute force solution. **You are not allowed to use goto label statements, exit statements. You are required to pass parameters to functions and not allowed to declare global or static variables.**)
   - Consistently employ coding conventions
   - Include internal documentation (i.e., comments)
3. Deadline for the project is the **8:00AM of December 5, 2022 (Monday)** via submission through **AnimoSpace**. After this time, submission facility is locked and thus no MP will be accepted anymore and this will result to a **0.0** for your machine project.
4. The following are the deliverables:

Checklist:
❒ Upload in AnimoSpace by clicking **Submit Assignment** on Machine Project and adding the following files: ❒ source code* ❒ test script**
❒ email the softcopies of everything as attachments to **YOUR own email address** on or before the deadline

Legend:

*****Source Code also includes the internal documentation. The **first few lines of the source code** should have the following declaration (in comment) **BEFORE** the introductory comment:

```
/*****************************************************************
*******
This is to certify that this project is my own work, based on my personal efforts in
studying and applying the concepts learned. I have constructed the functions and
their respective algorithms and corresponding code by myself. The program was
run, tested, and debugged by my own efforts. I further certify that I have not copied
in part or whole or otherwise plagiarized the work of other students and/or persons.
                                        <your full name>, DLSU ID#
<number>
*****************************************************************
*******/
```

Test Script should be in a table format, with header as shown below. There should be **at least 3 distinct test classes (as indicated in the description) **per function**. There is no need to create test scripts for functions that only perform displaying on screen.

Function Name	#	Test Description	Sample Input (either from the user or to the function)	Expected Result	Actual Result	P/F
getAreaTri	1	**base and height measurements are less than 1.**	**base = 0.25 height = 0.75**	**…**	**…**	
	2	:::				
	3					

Test descriptions are supposed to be unique and should indicate classes/groups of test cases on what is being tested. Given the function getAreaTri(), the following are 3 distinct classes of tests:

i.) testing with base and height values smaller than 1
ii.) testing with whole number values for base and height
iii.) testing with floating point number values for base and height, larger than 1.

The following test descriptions are incorrectly formed:

Too specific: testing with base containing 0.25 and height containing 0.75
Too general: testing if function can generate correct area of triangle
Not necessary -- since already defined in pre-condition: testing with base or height containing negative values

5. **MP Demo:** You will demonstrate your project on a specified schedule during the last weeks of classes. Being unable to show up on time during the demo or being unable to answer convincingly the questions during the demo will merit a grade of **0.0** for the **MP**. The project is initially evaluated via black box testing (i.e., based on output of running program). Thus, if the program does not compile successfully using gcc -Wall and execute in the command prompt, a grade of 0 for the project will be incurred. However, a fully working project does not ensure a perfect grade, as the implementation (i.e., correctness and compliance in code) is still checked.
6. Any requirement not fully implemented and instruction not followed will merit deductions.
7. This is an **individual project**. Working in collaboration, asking other people's help, and/or copying other people's work are considered as cheating. Cheating is punishable by a grade of **0.0** for CCPROG1 course, aside from which, a cheating case may be filed with the Discipline Office.
8. The above description of the program is the basic requirement. A maximum of 10 points will be given as bonus. Use of colors may not necessarily incur bonus points. Sample additional features could be:
   (a) use of arrays or dynamic lists. This may require you to read ahead or research on how to apply these properly to your project.

   Note that any additional feature not stated here may be added but **should not conflict with whatever instruction was given in the project specifications**. Bonus points are given upon the discretion of the teacher, based on the difficulty and applicability of the feature to the program.

Note that **bonus points can only be credited if all the basic requirements are fully met** (i.e., complete and no bugs).

HONESTY POLICY AND INTELLECTUAL PROPERTY RIGHTS

Honesty policy applies. Please take note that you are NOT allowed to borrow and/or copy-and-paste – in full or in part any existing related program code from the internet or other sources (such as printed materials like books, or source codes by other people that are not online). **You should develop your own codes from scratch by yourself.**