Small Program 5

COP3223C Introduction to Programming with C — Fall 2022

Dr. Andrew Steinberg

Due Date

The assignment is due on October 29th at 11:59pm EST via Webcourses. **Do not email the professor or TAs your submissions as they will not be accepted!** This assignment is accepted late up to 24 hours with a penalty. Please see the syllabus for more information on this. Make sure to submit on time to get potential full credit. Make sure to also take into consideration the uploading time. In the past, students who are working last minute on the assignment sometimes run into uploading issues where their Internet may run slow, resulting in late submissions. The timestamp Webcourses uses for your submission will be applied and will be the final say. Please do not email the instructor or TAs saying your Internet was running slow. If the time is off by a second of the due date, then the assignment is considered late. Plan accordingly!

Important! Read Carefully! Something new has been added!

This assignment contains a set of problems that are to be completed in **one C file**. You have learned about creating user-defined functions and why they are so beneficial to us programmers. For each problem in the assignment, you will create the definition of the user-defined function that is asked in the description. **If you do not create a user-defined function for each of the problems, then you will receive no credit for the problem.** Creating user-defined functions is good practice! You also must write the function prototypes! Missing function prototypes will result in points being deducted. Function prototypes are also good practice as well. The file must be named *smallprogram5_lastname_firstname.c*, where lastname and firstname is your last and first name (as registered in webcourses). For example Dr. Steinberg's file would be named *smallprogram5_Steinberg_Andrew.c*. Make sure to include the underscore character _. If your file is not named properly, points will be deducted. The script the graders use will pull your name from the file name properly. It is imperative you follow these steps so you can get your points! In this small programming assignment, you have two problems that involve use of FILE I/O. The necessary text files have been provided for you in Webcourses in the assignment page.

Testing on Eustis

It is your responsibility to test your code on Eustis. If you submit your assignment without testing on Eustis, you risk points being deducted for things that may behave differently on your operating system. Remember, you cannot dispute grades if your code didn't work properly on Eustis all because it worked on your machine. The Eustis environment gives the final say. Plan accordingly to test on Eustis!!

The Python Script File! Read Carefully! NEW INFORMATION!

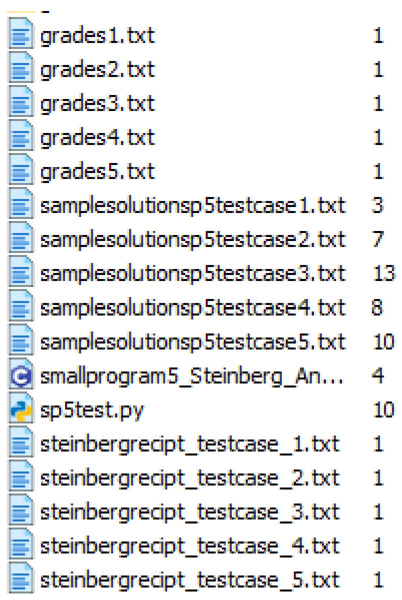
A python script has been provided for you to test your code with a sample output of Dr. Steinberg's solution. This script will check to make sure your output matches exactly with Dr. Steinberg's solution file as the graders are using this to grade your assignments. The script removes leading and trailing white space, but not white space in the actual text. If there is anything off with the output (including the expected answer), the script will say your output is not correct. This includes your output producing the correct answer, however there is something off with the output display. The script is going to run 5 unique scenarios for each problem (5 Test Cases). Each test case contains a different set of input values being used to ensure your code produces the correct answer. Back in your previous assignments, Dr. Steinberg would provide 1 sample solution that you would upload to Eustis. Now, there are 5 solution text files you are going to need to upload to Eustis. Before you test your program, your directory in Eustis should look something like this:

NEW INFO FOR FILE I/O In this assignment there are 2 FILE I/O problems. In order to properly test them with the script, you are going to need to upload 10 text files (for problems 2 and 3) that are going to be read OR compared from writing. Those files are:

1. grades1.txt
2. grades2.txt
3. grades3.txt
4. grades4.txt
5. grades5.txt
6. steinbergrecipt_testcase_1.txt
7. steinbergrecipt_testcase_2.txt
8. steinbergrecipt_testcase_3.txt
9. steinbergrecipt_testcase_4.txt
10. steinbergrecipt_testcase_5.txt

Your Eustis Directory should look something like this: If you have these files, you are ready to run the script. Use the following command to test your code with Dr. Steinberg's provided solution sample.

```
python3 sp5test.py
```



```
grades1.txt                        1
grades2.txt                        1
grades3.txt                        1
grades4.txt                        1
grades5.txt                        1
samplesolutionsp5testcase1.txt     3
samplesolutionsp5testcase2.txt     7
samplesolutionsp5testcase3.txt     13
samplesolutionsp5testcase4.txt     8
samplesolutionsp5testcase5.txt     10
smallprogram5_Steinberg_An...      4
sp5test.py                         10
steinbergrecipt_testcase_1.txt     1
steinbergrecipt_testcase_2.txt     1
steinbergrecipt_testcase_3.txt     1
steinbergrecipt_testcase_4.txt     1
steinbergrecipt_testcase_5.txt     1
```

Figure 1: Your setup for testing on Eustis.

If the script says your output is incorrect, checkout the sample text file that was generated (a new text file will be created from the script that contains YOUR output). If your numbers are off or different from Dr. Steinberg's, then that means there is something not right with code's logic and calculating the answer. However if your numbers match with Dr. Steinberg's solution, then that means there is extra/missing white space or newlines detected. Compare the text file generated by the script with solution text file line by line to find the missing/extra white space or newlines. Once you believe you found the error, rerun the script to see if the output matches.

The Rubric

Please see the assignment page for the established rubric on webcourses.

Comment Header

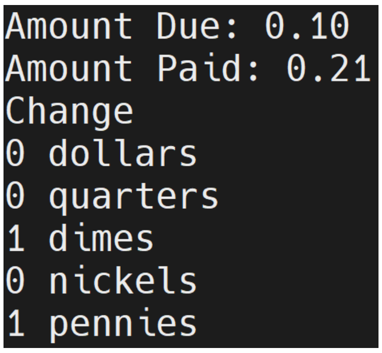
Make sure you place a comment header at the top of your C file. You will use single line comments to write your name, professor, course, and assignment. For example, Dr. Steinberg's header would be:

```
//Andrew Steinberg
//Dr. Steinberg
//COP3223C Section 1
//Small Program 5
```

Missing a comment header will result in point deductions!

Problem 1

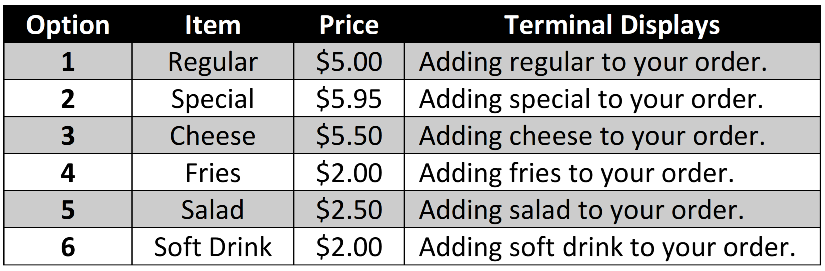
Write a user defined function called `change`. The user enters the amount paid and the amount due in the main function. The program determines how many dollars, quarters, dimes, nickels, and pennies should be given as change. The function parameter has two double reference parameters. The first parameter represents the amount paid and the second represents the amount due. Assume the user will always pay over the amount due. Pointers are required to receive full credit for the problem. If pointers aren't used, then no credit will be given. **IMPORTANT!** In this problem some of you may run into a round off error (due to loss of information). This is very normal to witness in the industry and it's good to learn how to deal with them. Some programmers will define a small ϵ value such as (0.00025) to deal with this round off error. There are other methods that exists. Make sure to talk to the TAs and ULAs if you are struggling with this problem. You do not need to worry about invalid input for this problem. The following figure shows the sample output for this problem.



```
Amount Due: 0.10
Amount Paid: 0.21
Change
0 dollars
0 quarters
1 dimes
0 nickels
1 pennies
```

Figure 2: Sample output for problem 1. Make sure it matches this for the script when testing.

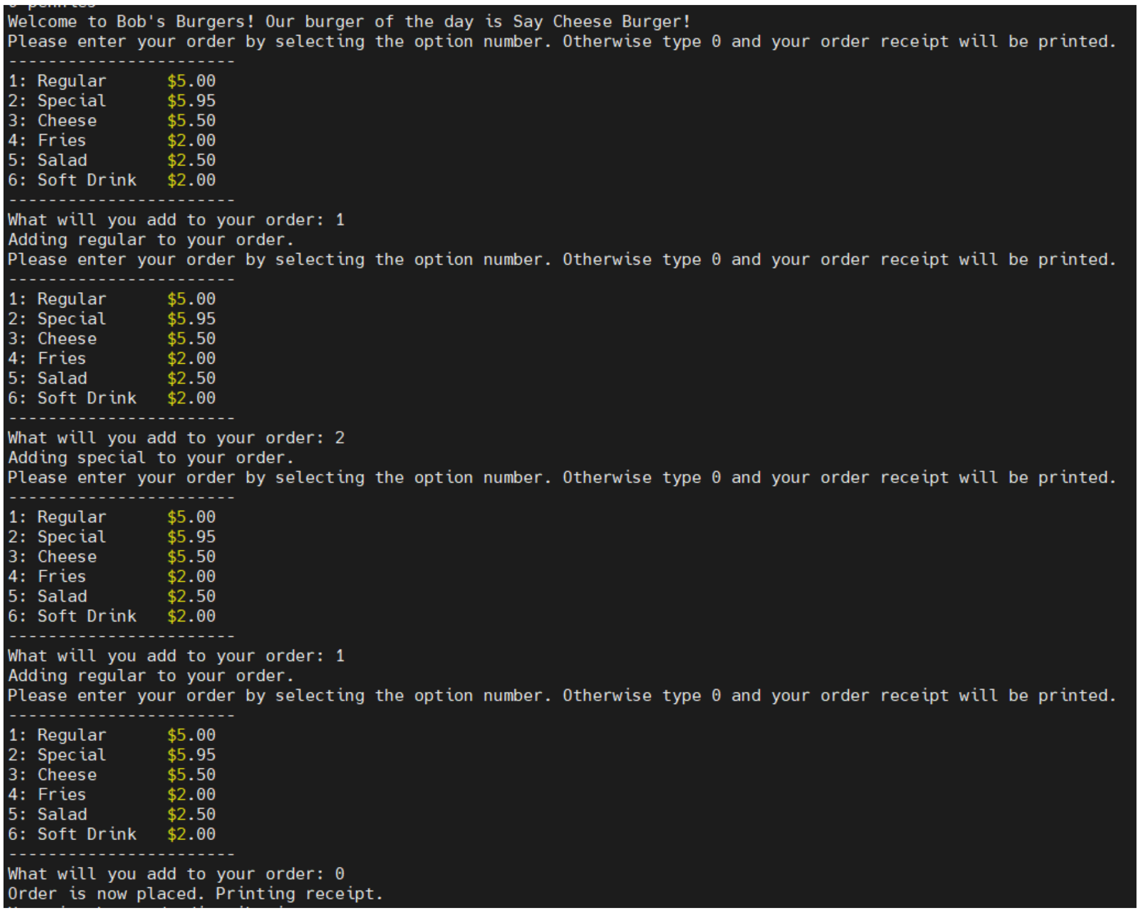
Problem 2



Option	Item	Price	Terminal Displays
1	Regular	$5.00	Adding regular to your order.
2	Special	$5.95	Adding special to your order.
3	Cheese	$5.50	Adding cheese to your order.
4	Fries	$2.00	Adding fries to your order.
5	Salad	$2.50	Adding salad to your order.
6	Soft Drink	$2.00	Adding soft drink to your order.

Figure 3: Table that shows option with associated item, price, and response in the terminal after user selects option.

You were currently hired to work at Bob's Burgers Restaurant for a temp contract position. Your position is helping the restaurant create receipts for customers who want a copy of their order transaction. Write a user defined function called `resterauntReceipt`. The function has no parameters and does not return anything. Inside the function, you are going to collect the order of the customers. This includes the number of each item along with the associated price. After the user finishes ordering, the function will print a receipt to a text file. The name of the receipt is a text file called `myreceipt.txt`. Use the above table in figure 3 to create the menu/option of selecting items to the order along with the response that is displayed to the terminal window. You do not need to worry about invalid options from being entered in this problem. Once the user is done entering their order, they must hit 0 to proceed with the receipt printing. When 0 is entered, the terminal will display the message `Order is now placed. Printing receipt.` This will cause the text file to be generated with the items selected along with the total cost. Check out the sample text file in webcourses to see the formatting of the text file.

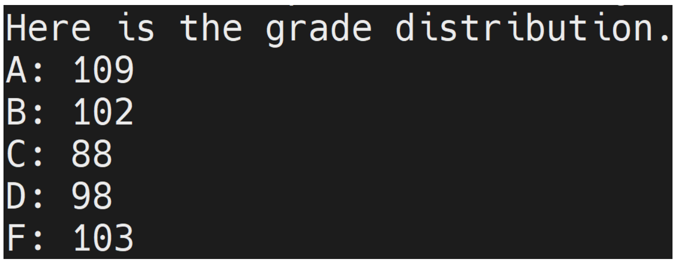


```
Welcome to Bob's Burgers! Our burger of the day is Say Cheese Burger!
Please enter your order by selecting the option number. Otherwise type 0 and your order receipt will be printed.
-----------------------
1: Regular      $5.00
2: Special      $5.95
3: Cheese       $5.50
4: Fries        $2.00
5: Salad        $2.50
6: Soft Drink   $2.00
-----------------------
What will you add to your order: 1
Adding regular to your order.
Please enter your order by selecting the option number. Otherwise type 0 and your order receipt will be printed.
-----------------------
1: Regular      $5.00
2: Special      $5.95
3: Cheese       $5.50
4: Fries        $2.00
5: Salad        $2.50
6: Soft Drink   $2.00
-----------------------
What will you add to your order: 2
Adding special to your order.
Please enter your order by selecting the option number. Otherwise type 0 and your order receipt will be printed.
-----------------------
1: Regular      $5.00
2: Special      $5.95
3: Cheese       $5.50
4: Fries        $2.00
5: Salad        $2.50
6: Soft Drink   $2.00
-----------------------
What will you add to your order: 1
Adding regular to your order.
Please enter your order by selecting the option number. Otherwise type 0 and your order receipt will be printed.
-----------------------
1: Regular      $5.00
2: Special      $5.95
3: Cheese       $5.50
4: Fries        $2.00
5: Salad        $2.50
6: Soft Drink   $2.00
-----------------------
What will you add to your order: 0
Order is now placed. Printing receipt.
```

Figure 4: Sample output for problem 2.

Problem 3

Dr. Steinberg needs your help! He just finished assigning final letter grades in his Microcomputers Applications course and wants to know the grade distribution. Grade distribution is simply seeing the number of the students that were awarded with the respective letter grade. In his Microcomputers Applications class, Dr. Steinberg doesn't use the plus/minus system. That means the only letter grades are A, B, C, D, and F in the Microcomputer Applications course. The registrar site downloads all the letters as a txt file called `grades.txt` (already provided in Webcourses). Each line in the file contains a letter grade that was awarded. Write a user defined function called `gradeDistribution` that takes no parameters. The function opens the text file of letter grades and reads each one line by line. After reading all the grades, the program will display the distribution of each grade assigned. The text file that you will read from must be in the same directory as your C source file. You do not need to worry about invalid input and each letter will be on its own line.

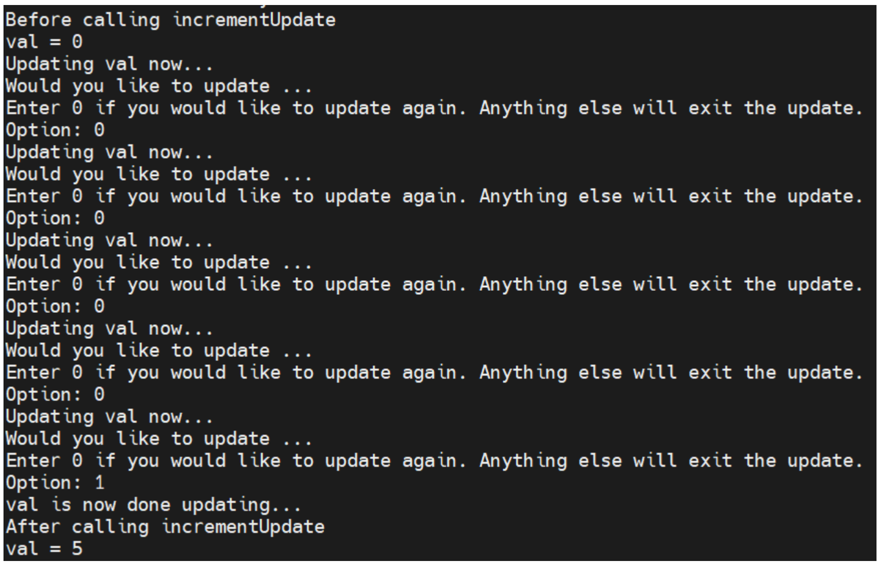


```
Here is the grade distribution.
A: 109
B: 102
C: 88
D: 98
F: 103
```

Figure 5: Sample output for problem 3.

Problem 4

Write a user defined function called `incrementUpdate`. The function takes one integer reference called `val`. The function updates the value stored in `val` by `1`. The user will be able to update `val` until the user enters another option. This other option will cause the user defined function to terminate. Display the value stored in `val` in the main function before and after the call. Inside the main the function, declare and initialize `val` to `0`. If pointers aren't used, then no credit will be given.



```
Before calling incrementUpdate
val = 0
Updating val now...
Would you like to update ...
Enter 0 if you would like to update again. Anything else will exit the update.
Option: 0
Updating val now...
Would you like to update ...
Enter 0 if you would like to update again. Anything else will exit the update.
Option: 0
Updating val now...
Would you like to update ...
Enter 0 if you would like to update again. Anything else will exit the update.
Option: 0
Updating val now...
Would you like to update ...
Enter 0 if you would like to update again. Anything else will exit the update.
Option: 0
Updating val now...
Would you like to update ...
Enter 0 if you would like to update again. Anything else will exit the update.
Option: 1
val is now done updating...
After calling incrementUpdate
val = 5
```

Figure 6: Sample output for problem 4. Make sure it matches this for the script when testing.