Programming Assignment - 1

CS200 - S2

Dr. Mian Muhammad Awais

Before starting the assignment, you should carefully read the following instructions:

- **Start the assignment early so that you can finish within the deadline. We have the following late submission policy: you are given a pool of total 5 grace days that you can utilize for all assignments with a 0% reduction in your marks. Remember that there will be no extensions after you have utilized all the grace days, so use them wisely across all assignments. Do not wait for the last minute to submit to avoid late submission due to connectivity issues.**

- **Read the questions thoroughly and plan out your solutions before starting to code.**

- **Please write neat, well indented, well commented code, and use self-describing variable names. This will really help the TA's to easily check your code and understand your logic.**
  Tip: You should use one format of variable naming and then stick to it. There are two types:
  1. **Camel Case: variableNamingConvention**
  2. **Snake Case: variable_naming_convention**

  You can google more instructions on how to write cleaner code. Look up on the internet for a command that clears the screen of the same user output window for a new part (especially for Q3), like system("cls").

Do	Don't
`func(num1, num2, num3, num4)`	`func(num1,num2,num3,num4)`
`int number = 3 + 5 - (6 * 4);` `cout << "hello" << "world" << endl;`	`int t=3+5-(6*4);` `cout<<"hello"<<"world"<<endl;`

`while(a != 0 && !(b || c))`
`{`
`  //code`
`}`
`OR`
`while(a != 0 && !(b || c)) {`
`  //code`
`}`
	`while(a!=0&&!(b||c)){//code}`
	`OR`
	`while(a!=0&&!(b||c))`
	`{`
	`//code`
	`}`

- For Q1 and Q2, you should submit a ***SINGLE*** .cpp file. Make a main () function and make calls to the respective functions from it. Remember to cout a heading and endlines for every question before its output is shown. Submit a separate .cpp file for Q3. You need to make a zip folder containing these TWO .cpp files, yourRollNumber_q1q2.cpp and yourRollNumber_q3.cpp, and name it as PA1_assignedTAName_rollNumber.zip.

- We expect your programs to work on all valid inputs, if it fails to work on any valid input, you will be awarded half marks for that part. The examples given under questions are NOT exhaustive, be sure to check all corner cases.

- You can choose to not use structs for Q1 and Q2 (but if you do, it's better), however it is mandatory to use them in Q3.

- You will be marked out of 10 for following the submission instructions properly [2] and writing clean code[8].

- In case you have any queries, please feel free to post them on Piazza and ask the TAs in office hours.

- Do not copy code from the internet, share or look at any other student's code. If in doubt, talk to sir or any TA.

Total Marks: 155

Question 1 [25]:

A. Write a *swap ()* function that only takes pointers to two integer variables as parameters and swaps the values in those variables using the pointers without creating any extra variables. The swapped values are displayed from the main(). [5]

B. Write a program that will take a *char* array as an input, and it reverses the positions of the *char* elements in the array. You are not allowed to make a new array. Use the bracket notation of arrays for this question eg: arr[j + k]. [5]
Example:

```
Input: [ 'C', 'S', '2', '0', '0']
Output: ['0', '0', '2', 'S', 'C']
```

C. Write a program that will ask the user to enter a two integer arrays of variable size (this means that the user can tell the program the size of each array), and then computes the intersection and union of those arrays.
Note: the arrays can be of different lengths and the intersection/union should have unique elements in sorted order. Use pointer notation of arrays for this question. Eg: *(arr+1) [10]
Output Example:

```
Enter size of array 1: 2
Element 0, array 1: 10
Element 1, array 1: 9

Enter size of array 2: 5
Element 0, array 2: 9
Element 1, array 2: 8
Element 2, array 2: 3
Element 3, array 2: 3
Element 4, array 2: 10

Union: {3, 8, 9 ,10}
Intersection: {9, 10}
```

D. Write a function named *searchInsertPos ()* that takes in three arguments: a sorted array, size of the array, and an integer number. It should return the position where the integer value is found. In case the number does not exist in that array it should return the index where it should have been if it were present in this sorted array. Use pointer notation of arrays for this question. [5]

Question 2 [25]:

A. Write a function *isPalindromeDelete* () that will take a string containing only alphanumeric characters that are in lowercase (if you think your logic requires you to use more than one argument, please go ahead). Your task is to see if the string becomes a palindrome if only one character is removed from anywhere in the string.[12]

Example:

```
Input: "pop", Output: true

Input: "pops", Output: true

Input: "acde", Output: false

Input: "", Output: true
```

B. In this task you will implement a function *targetSum ().* Argument: a sorted integer array, size of the array, and a target integer value. You have to find all combinations of two elements in the sorted array which sum up to the target value. When found, add the combinations into an array, and print the array. In the case where there is more than one combination, you may use the number 999 as a separator for the two or more combinations in the output array. Note: You are not allowed to use more than one loop. Use pointer notation of arrays to do this question. [13]

```
Input: [2, 4, 6, 9, 11], target = 11

Output: [2, 9]

Input: [-10, 10, 10, 20, 30, 40], target = 20

Output: [-10, 30, 999, 10, 10]

Input: [-1, 4, 6, 9, 10], target = 0

Output: []
```

Question 3 [95]:

In this task, you will implement a restaurant system using structs (using the dot notation). There is a single person managing a new restaurant and several potential customers will come to the restaurant.

Initially, there will be a main menu displayed on the console to enter either of the two modes: customer or owner.

Main Menu [3]:

- Press 1 for customer mode
- Press 2 for owner mode
- Press 3 to exit program

If customer is selected, the following options should be available for the customers[2]:

- Press 1 for new customer
- Press 2 for current customer

The details of what the above selections would do is specified below:

1. New customer

- The customer is prompted to enter their name (assume every person at the restaurant has a different name) [2]
- The customer is asked to select a seat position to be seated and after successful seating, the customer menu is displayed. The criteria for successful seating is stated below [3]
- **Select seat [2+2+2+2+2]**:
    - There is a single row of 10 seats available, each empty seat depicted with an 'o'
    - The customer will select the numbered seat they wish to sit at, and the occupied seat will then turn 'x' (the size of the seats array will remain the same)
    - A customer may mistakenly try to sit in an already occupied seat and you should make sure that isn't possible. (report the exact problem e.g. Error: seat taken, choose another seat)
    - You should also make sure that whatever seat the customer chooses isn't to the immediate left or immediate right of an occupied seat, keeping in view social distancing rules.
    - The customer should be asked to select a different seat until a successful seating is done

2. Current customer (already seated)

- they enter their name and if the named person is present in any of the seats, the customer menu is displayed, otherwise an error is displayed to show this person is not currently at the restaurant [3]

Customer Menu [2]

Press 1 to View food menu
Press 2 to Place order
Press 3 to View order
Press 4 to Pay bill
Press 5 to Leave
Press 6 to Return to Main menu

The details of what should happen when any of the options are selected are explained as follows:

1. View food menu [10]:
    - This will simply display the current menu of the restaurant in the following format: (Serial no Dish name Price)

Hint: There should be an array of struct defined for menu items. There can be at most 10 items on the menu. This would be the same for both owner and customer.

- At initialization, the menu should have 4 items already (of your choice) (This is to allow space for adding more food items later)

2. Place order [10]:
    - The customer will place their order by entering the serial number of the food which will place the order (food item name) on the list for the owner to view as well.
    - For simplicity's sake, assume every customer only orders one item
3. View order [5]:
    - This will display the name of the item of food ordered by the customer
4. Pay bill [8 + 2]:
    - If a person selects to pay the bill, **the person's order is removed from the list of current orders kept by the restaurant** and the **customer is regarded as being served by the restaurant** and the **amount of the bill is added to the earnings of the restaurant**.
    - Make sure that a person is not able to pay if their order was empty
5. Leave [5]:
    - A customer must only be able to leave if they have no amount due
    - Their seat should be vacated after they leave

If owner mode is selected from the Main Menu, the following owner menu should be displayed on the console:

Owner Menu [3]

Press 1 to Update food menu
Press 2 to View food menu
Press 3 to View number of customers served
Press 4 to View orders received (names of food items)
Press 5 to Amount earned
Press 6 to Offer discount
Press 7 to Return to Main menu

The details of what should happen when any of the options are selected are explained as follows:

a. Update food menu [10]:
    - The owner would like to make the small café bigger so they would like to **add more food items to the menu but does not intend to get rid of any item once added on the menu card.**
    - The popularity of a dish may warrant changing its price so there should be an option to adjust the price of any item on the menu individually.
    - These two options should be displayed to the owner to select from
b. View number of customers served [2]:
    - This should simply display the total number of customers served
c. Orders received [5]:
    - The orders of customers who have not paid for their food yet are displayed
    - Assume there are no more than 10 orders made
d. Amount earned [5]:
    - This should display the total earnings the owner has made at the restaurant
e. Offer Discount [5]:
    - The owner would enter the percentage amount to offer discount which would apply to all items on the restaurant menu.