SENG1000 – C/C++ PROGRAMMING

MAJOR ASSIGNMENT 3 -USING ARRAYS AND C++ STRINGS/INPUT

OVERVIEW

Write a program that calculates hotel bills for a meeting.

GENERAL COURSE OBJECTIVES ADDRESSED IN THIS ASSIGNMENT

- Declare and use arrays, with initial data.
- Pass arrays as parameters.
- Use the C++ string class.
- Use streams for simple standard input and output.
- Use appropriate scope and style.
- Use arithmetic operators.
- Use if-else statements.
- Use a loop.
- Use pre-existing functions.
- Split your code into modular functions.
- Make appropriate design decisions.
- Follow stated requirements.

ACADEMIC INTEGRITY AND LATE PENALTIES

- Link to Academic Integrity Information
- Link to Late Policy

EVALUATION

- The evaluation of this assignment will be done as detailed in the Marking lecture from Week 2.

PREPARATION

- View Week 4 and 5 videos.

REQUIREMENTS

ARRAY DECLARATION

- Declare an array of doubles. The name of the constant must adhere to the SET Coding Standards. It must be a const array (i.e. const should precede the data type).
- This array must be declared in main().
- This array must have the following values (in order):
    - 127.50
    - 128.25
    - 130.50
    - 150
    - 150
    - 162.50

- Each element of the array represents the room rate on a particular night, starting on Sunday. So, the rate for Sunday is 127.50, the rate for Monday is 128.25, etc.

THE ATTENDEES:

- Up to four people are renting hotel rooms in order to attend a meeting.
- The meeting is on Thursday. However, each person has other tasks that they can do once they are in town so they might arrive earlier or leave later.
    - The week starts on Sunday (earliest check in) and ends on Saturday (latest check out).
    - In the case that any of the following constraints are violated, display an error message and skip this person.
        - They must check in Thursday at the latest.
        - They must check out Thursday at the earliest.
        - They cannot check in and check out on the same day (i.e. they need to stay in the hotel for at least one day).
- You must have a loop in main() that prompts the user for each of the people. The user must enter (on separate lines):
    - the person's name (which could have multiple parts; e.g. "Fred Flintstone")
        - if the name is empty (i.e. ""), display an error message and skip this person
    - the day in which they are check-ing in to the hotel
        - all days are strings
        - "sun" represents Sunday, which is the first day in which they can check into the hotel. The other day abbreviations are "mon", "tue", "wed", "thu", "fri", "sat", accordingly.
    - the day in which they are check-ing out of the hotel (the day input is similar to above; Saturday is the last day in which they can check out)
- All prompts for user input should **not** move the cursor to a new line. This would be a more natural user interface, in which the input is immediately to the right of the prompt.
- For each person, display their name and their total room cost. Then display a blank line. This is the only time when there should be a blank line in the output.
- At the end, display the total costs for the entire group.

CONVERTING DAYS TO NUMBERS

- In the array containing the room rates, the first value corresponds with the rate for Sunday. Each subsequent value corresponds with the next day of the week.
- You must create a function called dayToIndex(). It converts from the user input to a useful array index. It is called from main().
    - It must take exactly one parameter: a string indicating what day was entered by the user.
    - It must return an int indicating the day that was entered (0 represents Sunday, 1 is Monday, etc. to 6 being Saturday) or, in the case of the user not entering a valid day string, -1 indicating an error.

    - The -1 error value must be represented by a global constant. It must go before any function definitions in your code. The constant's name must adhere to the SET Coding Standards.
    - dayToIndex() must do no output.
    - The comparison is case-sensitive (e.g. "mon" represents Monday but "Mon" is an error). This is intended to make it easier for you.

CALCULATING ROOM COST:

- In order to calculate the cost of the room for a person, you must create a function called calculateCostOfRoom(). It is called from main().
    - It must take exactly these parameters, in order:
        - the array of room rates
        - the day of check-in, as an index into the array
        - the day of check-out, as an index into the array
    - It must return the total cost for the room, as a double.
    - It must do no output.
    - There is no need for error-check-ing in this function.

ERROR HANDLING

- There are five errors that you need to handle in this program:
    - Missing name
    - Invalid check-in day
    - Invalid check-out day
    - Invalid length of stay
    - Missing the meeting
- All output for the error messages must be done in main().
- Do your error check-ing as soon as you have the data to check. What that implies is the following:
    - if they are missing a name, display the error message and don't get the check-in or check-out days or display any cost
    - if the check-in day is invalid, display the error message and don't get the check-out day or display any cost
- Any of the five errors result in skipping the person (but do **not** quit the program).
    - e.g. if two people have errors associated with them, only the other two people with valid data are attending the meeting

INPUT AND OUTPUT

- All input must be done using std::getline().
    - You can use the std:: prefix or you can omit it (if you use "using namespace std;").
- All output must be done using cout.
- All strings must be C++ string objects, not C-style strings.

- The only blank lines should be present after each person is completed. Normal output should fit within the normal Visual Studio console window. Refer to the sample executable as a model.

COMMENTING AND STYLE:

- You must have a header comment similar to that found in the SET Coding Standards or the Course Notes.
- You must have function comments for all functions except main().
- You must have inline comments that adhere to the two main principles for good inline comments (covered in the Commenting lecture and Code Complete).
- Adhere to the principles covered in the Style lecture.
- The above points will apply identically for subsequent assignments as well.

OTHER REQUIREMENTS:

- It is important that you meet the requirements when provided for defining functions exactly, as (in industry) they often are provided as definitions as part of a larger system. Deviating from the requirements can break the system.
- Use the SET Coding Standards (found on eConestoga) that are relevant.
- For all input and string handling, the program must only use the C++ features that were taught in the course.
- You can create more functions if you would like but there isn't any need to do so.
- The program must not use goto (this requirement holds for all subsequent assignments).
- All variables must be declared within functions (i.e. it must not use global variables (covered in the Scope and Style lecture)).
- Appropriate programming style as discussed in lecture and in the Course Notes must be used.
- **Do not have any input or output except as required by this assignment.**
- It is assumed that you will adhere to all course requirements detailed in the Course Notes readings so far in the course. **This requirement holds for all subsequent assignments.**
- Do not clear the screen (this requirement is true for all assignments).

CHECKLIST REQUIREMENTS

- Create a requirements checklist. This should contain the specific requirements from this assignment as well as any relevant requirements that have been covered in lecture or that are found in the SET Coding Standards or SET Submission Standards. Do it in whatever form you wish. Hand in your completed checklist in PDF form as checklist.pdf. Not having this checklist will result in a cap of 80 on your mark.

FILE NAMING REQUIREMENTS

- You must call your source file m3.cpp.
- You must call your checklist checklist.pdf.

SUBMISSION REQUIREMENTS

- Do not hand in any other source files besides those mentioned in the File Naming Requirements.
- Follow the instructions in the SET Submission Standards and the lecture on Submitting Assignments to submit your program. Submit both files in one submission to the correct Assignment folder.
- Once you have submitted your files, make sure that you've received the eConestoga e-mail confirming your submission. Do not submit that e-mail (simply keep it for your own records until you get your mark).

ADDITIONAL INFORMATION

- Just to be absolutely clear, they do not have to pay for the day of check-out (i.e. if they check in on Wednesday and check out on Thursday, they pay only for Wednesday).
- It should be noted that, as a general principle, you should keep input and output out of functions that do calculations or conversions. That application of a principle of modular programming will help with reusability (since the function does just one thing but does it concisely and well).
- In assignments in general, if you are required to display output but the text of the message is not specified in the requirements, you can choose what that wording is. Be reasonable in your choice and do not insult the user (you'd think that this would be unnecessary to say but it's not).
- The source code for the sample solution was about 100 lines, excluding comments. You don't have to adhere to this but I'm just letting you know in case you're doing way too little (so you're probably missing something) or way too much (so you're overcomplicating it).