CS201 – Fall 2021-2022

Take-Home Exam 2
– Budget Problem –
Due November 3rd, Wednesday, 23:55 (Sharp Deadline)

Introduction

The aim of this take-home exam is to make students comfortable with parametric functions (return types, parameters, code reuse, pass by value, pass by reference etc.) in C++ as well as using computational approaches to solve mathematical questions. You will practice on modularity and code reuse. It is possible to finish this homework without using functions, but it is a **must** to use them.

Description

In this homework, you will solve a budget problem using a C++ program. Given prices of 3 objects, you will take 3 quantities for these objects and a total budget as inputs and without knowing which quantity refers to which object you need to calculate the most expensive permutation of these objects that does not exceed the budget.

You **cannot** write all of your code under the main function, i.e. you **must** write user-defined functions and use them.

Your take-home exams will be automatically graded using GradeChecker, so it is very important to satisfy the exact same output given in the sample runs. You can utilize GradeChecker (http://learnt.sabanciuniv.edu/GradeChecker/) to check whether your implementation is working in the expected way. To be able to use GradeChecker, you should upload all of your files used in the take-home exam (**only** *your_main_cpp* file for this take-home exam). Additionally, you should submit all of your files to SUCourse (**only** *your_main_cpp* file for this take-home exam) **without zipping** them. **Just a reminder, you will see a character ¶ which refers to a newline in your expected output.**

The name of your main source (cpp) file should be in the expected format: "SUCourseUsername_THEnumber.cpp" (all lowercase letters, e.g. gulsend_THE2.cpp). Please check the submission procedures of the take-home exam, which are listed at the end of this document.

Inputs, Flow of the Program and Outputs

The inputs of the program and their order are explained below. It is extremely important to follow this order with the same characters since we automatically process your programs. ***Thus, your work will be graded as 0 unless the order is entirely correct***. Please see the "Sample Runs" section for some examples.

The prompts of the input statements to be used has to be exactly the same as the prompts of the "Sample Runs".

The program starts by asking the user's total budget in TL. This number will indicate how much a user can spend and therefore, it cannot be negative. You should do an input check on the budget such that if the user enters a negative number, the program should display an error message saying, "`Budget cannot be negative.`"

Then, your program should ask the user quantities of 3 objects. These 3 values must be positive. If one of them is entered as nonpositive, you should display a message "`All quantities must be positive.`" So, you should perform another input check on quantities.

After inputs pass all checks, you will be able to calculate the budget problem. As a result, you need to display the user's remaining money. If the user cannot afford given quantities in any way, you should prompt a message "`You cannot afford any of the permutations with these quantities and budget`".

After finishing the first user, your program needs to do everything one more time. In other words, your program will take inputs of the first user, calculate, and print an appropriate message, then asks for the second user's inputs and repeat the calculations. When you complete the task twice, prompt a goodbye message and finish your program. Please see the "Sample Runs" section for some examples.

Use of Functions

You are expected to implement a total of seven user-defined functions (other than main).

Function1 (returns bool type): This function is a parametric boolean function that performs input checks for the validity of the user's budget**.** It will take an integer variable as parameter, check if it is negative, and return true or false accordingly.

Function2 (returns bool type): This function is also a parametric boolean function. It will perform the input check for the validity of given quantities. It will

take three integers as parameters, check if they are **all** positive and return true and return false otherwise.

Function3 (returns int type): You will write a function that takes three integers (quantities) as parameters, calculates the total price with these given quantities and returns it. For this function, please take the prices of objects as **5 TL, 10 TL and 15 TL respectively.**

Function4 (returns int type): You will write another function that will perform a conditional maximum operation. It will take 3 integers as parameters: *previous_max_price*, *current_price* and *budget*. It will return *previous_max_price* if

- *previous_max_price* > *current_price*

or

- *current_price* > *budget*

and returns *current_price* otherwise.

Function5 (returns nothing, void): You will write a function that utilizes *Function3* and *Function4*. It takes three quantities, *budget* and *max_price* (total of 5) as parameters (all of them are `int`). All parameters except *max_price* should be passed by value, and *max_price* should be passed by reference. It first calculates the *current_price* using *Function3* and 3 quantity parameters. Then, it will call *Function4* using *max_price*, *current_price* and *budget*, and update *max_price*. Since it is a void function, no return is needed. However, you will use a reference parameter (*max_price*) to apply the same idea.

Function6 (returns int type): This function will utilize *Function5*. It will take 3 quantities and *budget* as parameters (all of them are `int`). Then it will use *Function5* to evaluate all the permutations of quantities. In other words, you will call *Function5* six times for each permutation and extract the maximum priced value that does not exceed the budget. After that it will return *max_price*.

Function7 (returns nothing, void): This function will be your runner function. Since you will do everything twice, you need to modularize your code so that when you call this function in main twice, it will repeat everything you implemented.

IMPORTANT NOTE: Please give appropriate names (explaining what it is doing) to your functions.

IMPORTANT!

If your code does not compile, then you will get **zero**. Please be careful about this and double check your code before submission.

VERY IMPORTANT!

Your programs will be compiled, executed and evaluated automatically; therefore you should definitely follow the rules for prompts, inputs and outputs. See **Sample Runs** section for some examples.

- **Order of inputs and outputs** must be in the mentioned format.

Following these rules is crucial for grading, otherwise our software will not be able to process your outputs and you will lose some points in the best scenario.

Sample Runs

Below, we provide some sample runs of the program that you will develop. The *italic* and **bold** phrases are inputs taken from the user. You have to display the required information in the same order and with the same words and characters as below.

Sample Run 1

```
Please enter your budget: -1150
Budget cannot be negative.

Please enter your budget: 150
Please enter three quantities: 5 4 4
You have 15 liras left.

Goodbye!
```

Sample Run 2

```
Please enter your budget: 50
Please enter three quantities: -1 -5 -2
All quantities must be positive.

Please enter your budget: 150
```

```
Please enter three quantities: 0 2 3
All quantities must be positive.

Goodbye!
```

Sample Run 3

```
Please enter your budget: 5
Please enter three quantities: 2 2 2
You cannot afford any of the permutations with these quantities and
budget.

Please enter your budget: -10
Budget cannot be negative.

Goodbye!
```

Sample Run 4

```
Please enter your budget: 100
Please enter three quantities: 1 2 3
You have 30 liras left.

Please enter your budget: 0
Please enter three quantities: 1 1 1
You cannot afford any of the permutations with these quantities and
budget.

Goodbye!
```

Sample Run 5

```
Please enter your budget: 120
Please enter three quantities: 5 4 3
You have 5 liras left.

Please enter your budget: 120
Please enter three quantities: 3 5 4
You have 5 liras left.

Goodbye!
```

General Rules and Guidelines about Homeworks

The following rules and guidelines will be applicable to all take-home exams, unless otherwise noted.

- **How to get help?**

You can use GradeChecker (http://learnt.sabanciuniv.edu/GradeChecker/) to check your expected grade. Just a reminder, you will see a character ¶ which refers to a newline in your expected output.

You may ask questions to TAs (Teaching Assistants) or LAs (Learning Assistants) of CS201. Office hours of TAs/LAs are at the course website.

- **What and Where to Submit**

You should prepare (or at least test) your program using MS Visual Studio 2012 C++ (Windows users) or using Xcode (macOS users).

It'd be a good idea to write your name and last name in the program (as a comment line of course). Do not use any Turkish characters anywhere in your code (not even in comment parts). If your name and last name is "İnanç Arın", and if you want to write it as comment; then you must type it as follows:

// Baris Altop

Submission guidelines are below. Since the grading process will be automatic, students are expected to strictly follow these guidelines. If you do not follow these guidelines, your grade will be 0.

- Name your submission file as follows:
    - Use only English alphabet letters, digits or underscore in the file names. Do not use blank, Turkish characters or any other special symbols or characters.
    - Name your cpp file that contains your program as follows:

"**SUCourseUsername_THEnumber.cpp**"

    - Your SUCourse user name is actually your SUNet username, which is used for checking sabanciuniv emails. Do NOT use any spaces, non-ASCII and Turkish characters in the file name (**use only lowercase letters**). For example, if your SUCourse username is "**altop**", then the file name should be: **altop_THE2*.cpp*** (please only use lowercase letters).
    - Do not add any other character or phrase to the file name.

- Please make sure that this file is the latest version of your take-home exam program.
- Submit your work **through SUCourse only**! You can use GradeChecker only to see if your program can produce the correct outputs both in the correct order and in the correct format. It will not be considered as the official submission. You must submit your work to SUCourse. You will receive no credits if you submit by any other means (email, paper, etc.).
- If you would like to resubmit your work, you should first remove the existing file(s). This step is very important. If you do not delete the old file(s), we will receive both files and the old one may be graded.

- Grading, Review and Objections

Be careful about the automatic grading: Your programs will be graded using an automated system. Therefore, you should follow the guidelines on the input and output order. Moreover, you should also use the same text as given in the "Sample Runs" section. Otherwise, the automated grading process will fail for your take-home exam, and you may get a zero, or in the best scenario, you will lose points.

Grading:

- There is NO late submission. You need to submit your take-home exam before the deadline. Please be careful that SUCourse time and your computer time may have 1-2 minutes differences. You need to take this time difference into consideration.
- Successful submission is one of the requirements of the take-home exam. If, for some reason, you cannot successfully submit your take-home exam and we cannot grade it, your grade will be 0.
- If your code does not work because of a syntax error, then we cannot grade it; and thus, your grade will be 0.
- Please submit your **own** work only. It is really easy to find "similar" programs!
- Plagiarism will not be tolerated. Please check our plagiarism policy given in the Syllabus or on the course website.

Plagiarism will not be tolerated!

Grade announcements: Grades will be posted in SUCourse, and you will get an Announcement at the same time. You will find the grading policy and test cases in that announcement.

Grade objections: It is your right to object to your grade if you think there is a problem, but before making an objection please try the steps below and if you still think there is a problem, contact the TA that graded your take-home exam from the email address provided in the comment section of your announced take-home exam grade or attend the specified objection hour in your grade announcement.

- Check the comment section in the take-home exam tab to see the problem with your take-home exam.
- Download the file you submitted to SUCourse and try to compile it.
- Check the test cases in the announcement and try them with your code.
- Compare your results with the given results in the announcement.

Good Luck!
Berker Demirel & Gülşen Demiröz & Barış Altop