# Sabancı University

Faculty of Engineering and Natural Sciences
CS204 Advanced Programming
Fall 2021

## Homework 6 – Bitwise operations with GUI

Due: 28/12/2021 (Tuesday), 12:30 pm (around noon)

**PLEASE NOTE:**

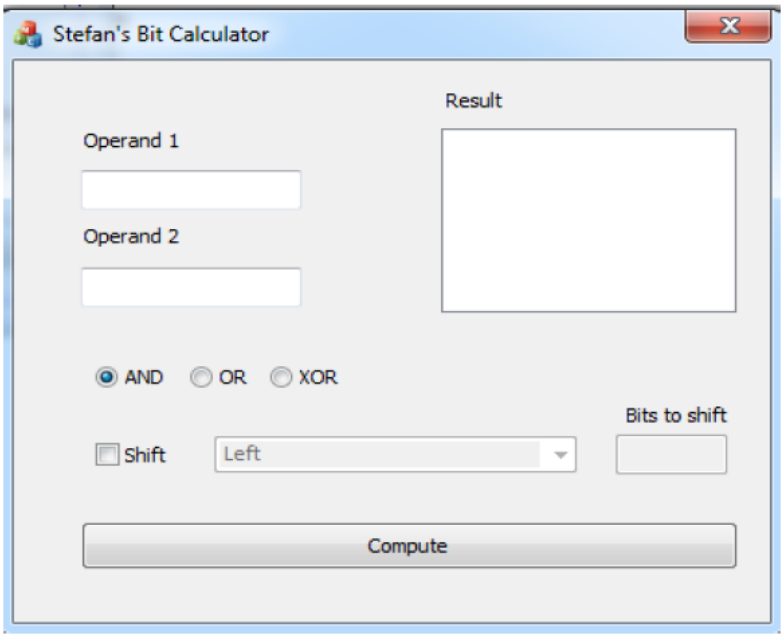
**Your program should be a robust one such that you have to consider all relevant user mistakes and extreme cases; you are expected to take actions accordingly!**

**You can NOT collaborate with your friends and discuss solutions. You have to write down the code on your own. Plagiarism will not be tolerated!**

### Introduction

In this homework, you are asked to implement a simple bit calculator with a Graphical User Interface (GUI). The calculator accepts as input two unsigned integers and based on the selected operation - bitwise AND, OR, XOR - the result is computed (again as an unsigned integer). Moreover, the bits of the operands might be shifted left or right by a user-specified number of bits before the selected operation. We will explain the details of the GUI design and the work flow hereafter.

### GUI



In the figure above, you can see the default values that should be displayed when the calculator is started: **"AND" is initially selected** as operator and **"Left" as shift direction**; moreover, the shift direction and corresponding field to enter the number of bits to be shifted are both **disabled**. These initial conditions must be held automatically by your program without user intervention.

The title of the application must be "*Your name*'s Bit Calculator".

All GUI elements that you can see are available as MFC elements. We now list their types and what each element should do.

The labels "Operand 1", "Operand 2", "Result" and "Bits to shift" are StaticText objects. The boxes under the labels "Operand 1", "Operand 2", and "Bits to shift" are EditControl objects in order to enter unsigned integers as input. The large box below the label "Result" is a ListBox object, which is used to display results of the computations and error messages. Each result/error message occupies exactly one line and if the text exceeds the boundaries of the ListBox object, a horizontal scrollbar must be displayed to allow users to read the whole line. The box is relatively small on purpose because this enforces the use of a horizontal scrollbar. *(Hint: see the example that we developed in the lab to enable horizontal and vertical scroll bars. Please recall that you should also write some code to enable a horizontal scroll bar)*.

"AND", "OR", and "XOR" are the bitwise operators and the user should choose one of them. They are all represented as RadioButton objects. Only one operator may be used for an operation, meaning no two or more operators can be selected at the same time. Moreover, initially "AND" must be selected automatically by your program when it is started.

"Shift" is a CheckBox object. Whenever it is selected, the corresponding ComboList object, that contains the options "Left" and "Right" to indicate the shift direction, and the EditControl object under the label "Bits to shift" must be enabled. Whenever "Shift" is deselected, both corresponding ComboList and EditControl objects must be disabled.

When the user clicks on the Button object named "Compute", the selected bitwise operator must be applied to Operand 1 and Operand 2 and the resulting unsigned integer must be displayed in the ListBox object labeled "Result". If the user selects "Shift" and specifies the direction and number of bits for shifting, both inputs Operand 1 and Operand 2 must first be shifted accordingly before the operator is applied to the operands. After the result is computed, it should be displayed directly without shifting again.

**Work Flow**

A user enters two operands as unsigned decimal integers, selects a bitwise operator, and optionally might want to shift the operands before the computation takes place. If the user decides to shift the operands by a specified number of bits either left or right, this operation is applied to the operands BEFORE the operator is applied to the operands.

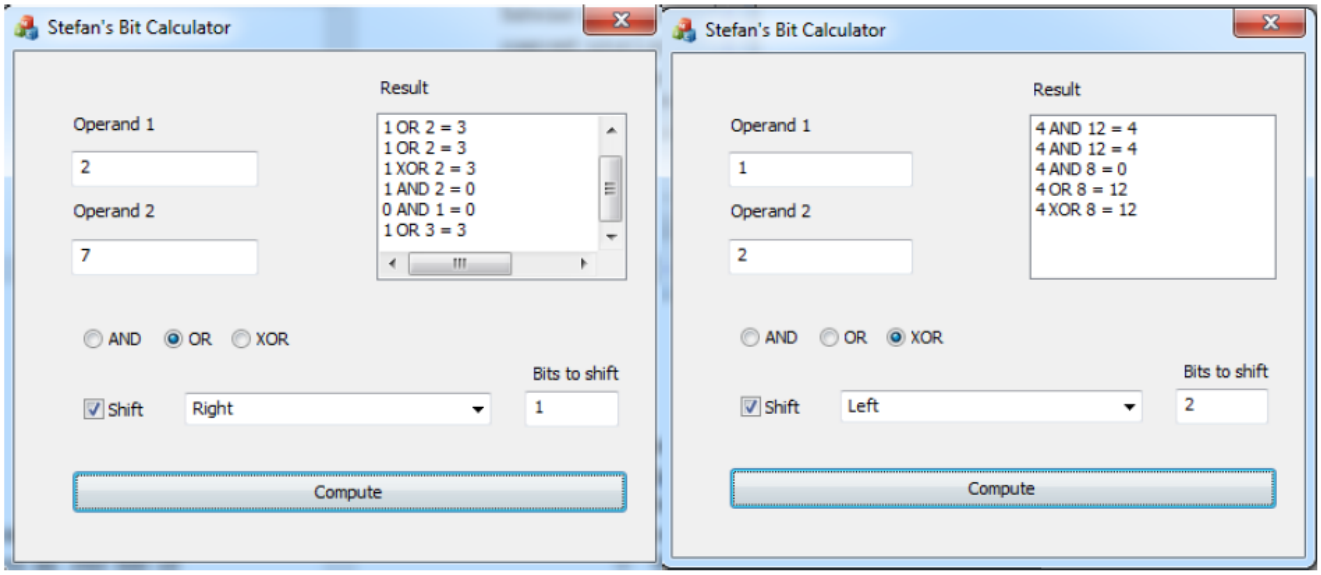
Important: users make mistakes. They might forget to enter one or both operands, or they do not specify the number of bits to shift. Thus, you must check these special cases. If at least one operand is missing, you should write in the "Result" box "At least one operand was not entered.". If the user forgot to specify the bits to shift, despite selecting "Shift", you must output "Specify how many bits you want to shift" instead of any result. If both inputs are missing, either one of the two error messages may be displayed. Since we use default values for the shift direction and the operator, the user cannot leave these inputs blank; so no checks are necessary for shift direction and operator.

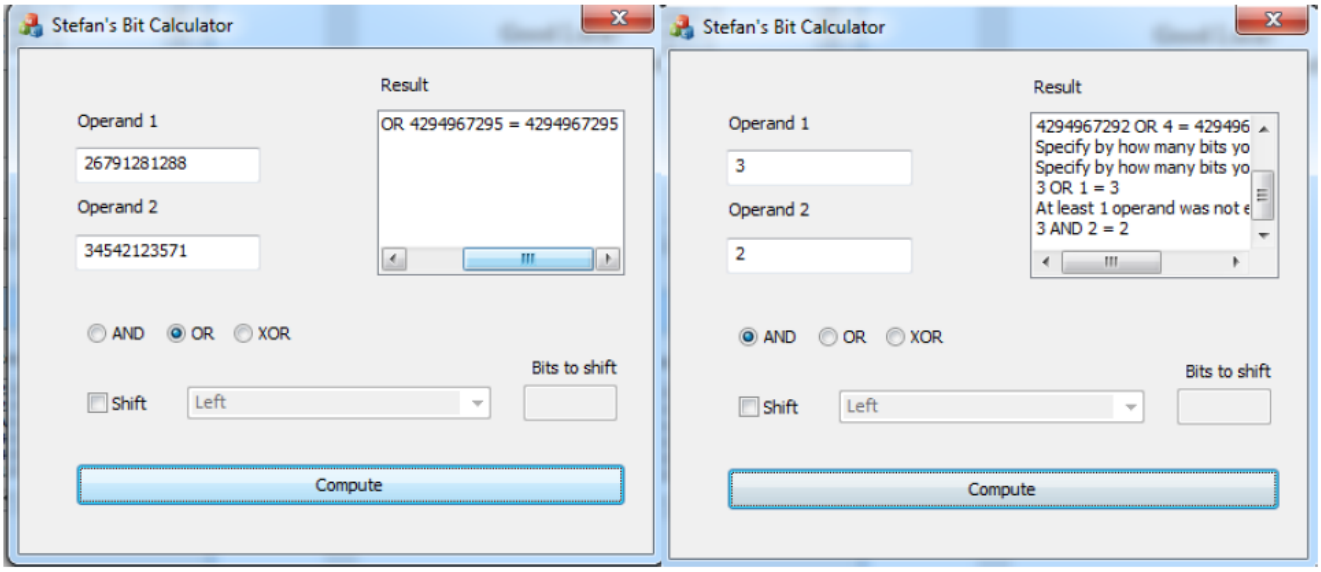
There are some input check cases that you can ignore. For example, if numbers entered as operands are larger than MAX_UINT, you do not have to worry about it; this causes overflow and the result will be incorrect, but this is OK for the homework. Another example case that you can ignore is that when you enter non-digits as the operands and number of bits to shift, the CString conversion function that you will use makes them zero. This is also OK.

**Remark**: since we use bitwise operations on the data, for all inputs unsigned int should be used. Since the data that you read via EditControl objects are of type CString, you must convert CString to unsigned int before (optionally) shifting the operands and before applying the bitwise operator. Similarly, since you can only display CString in ListBox object, you must convert unsigned int values to CString as well. The ways of doing these conversions are explained in **GUI_document.pdf** file given together with Lab12 material on SUCourse.

**Sample Runs**

Some sample snapshots are given below. However, the homework is better understood if you run the implemented version. In this homework zip package, we provide you an .exe file of the bit calculator that we developed. You can play with this sample program and see how the program operates.




**ATTENTION**

**Please see previous homeworks for the general rules that you have to follow**.

Since this time you will have the final product (application) in package with the HW6 document named as **hw6demo.exe**, if you have any questions, you can just check the behavior of the provided application and make sure that the final product of your code behaves similarly.

The submission deadline is 28.12.2021 (Tuesday), 12.30 pm. You can submit till 29.12.2021 (Wednesday), 12.30 pm with a penalty of 10 points.

Submission rules are similar to the previous homeworks, but there are some special considerations that you have to be very careful since this is a GUI (MFC) application.

- Use the same naming convention as in the previous homework assignments.
- Since this is not a console application, adding an existing cpp to a new project does not work. You have to develop your GUI application from scratch using the methods explained in the labs. Please check out the video recitation link of Lab12 link at SUCourse and old video recitation links at the recitation materials of Week 12 for more basic info on how GUI works.
- There are some other very important project files in addition to the cpp and header files. Thus please pay extreme attention that all project files exist in your zip file.
- Since the entire project would be very large (a couple of 100s of Mbytes), you will delete some files and folder while submitting. These are:
    - o the large file with .sdf extension
    - o the folder named ipch
    - o all debug and release folders
- These files and folders, which are deleted while submitting, regenerate themselves when you open the solution again. Thus, there is no harm not to submit those. However, please make sure that without those files, your solution opens up and your program compiles and runs correctly.
- As in the other homework assignments, correct submission is part of your duty; if we cannot run your program, we cannot grade it.

Good Luck!

CS204 Team!