İSTANBUL TEKNİK ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM FAKÜLTESİ
Date: 8.12.2020
Assoc.Prof.Dr.Tolga Ovatman
Assoc.Prof.Dr.Gülşen Eryiğit
Asst.Prof.Dr.Mehmet Baysan

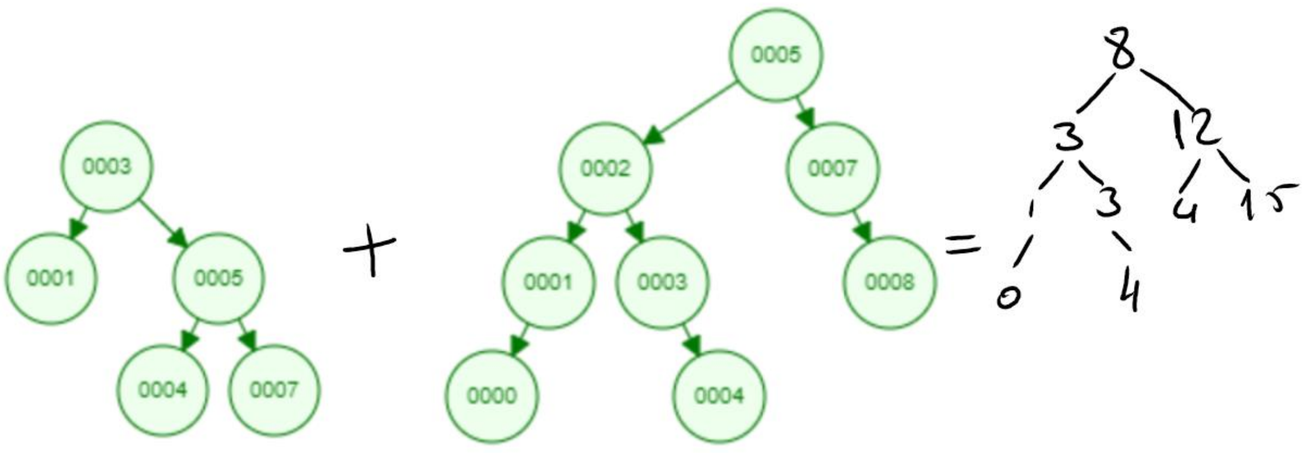
Student Id	**:**	
Name Surname	**:**	
Signature	**:**	

DATA STRUCTURES MIDTERM

Question	**1**	**2**	**3**	**Total**
Point				

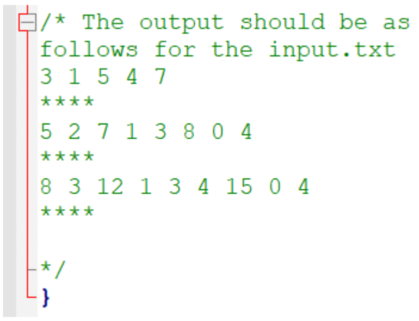
This exam will be an online exam. You are supposed to prepare your solutions as RUNNING codes. All the exam questions will also be provided as running functions. You are supposed to run and change them by using a C++ compiler, and upload the answers as separate files (as .cpp and/or .h) to Ninova. Please be careful to upload the answers for relevant questions. Upload each of your answers as you complete them.

Question 1. (35 pts) Assuming that you are given two binary search trees, whose data structures and creation methods are provided from Ninova, you are asked to write a recursive mergeBST function which will take two binary search trees as arguments, traverse them with a traversal method up to your choice and return a new binary tree (not necessarily a BST) which will be the sum of the input trees' elements occurring at the same tree position (i.e., whose nodes are superimposed sums of the nodes of the two trees). The original trees should stay the same; you are not allowed to change them or use their nodes directly in the construction of the new tree. In other words, the resulting binary tree should be constructed from scratch with its own nodes. The signature of the method and the main function are provided from Ninova. You are NOT allowed to change the main function and the existing methods. Please only fill in the body part of the mergeBST method. Writing extra methods is not allowed. An example is given below. The recursive method should be as short as possible (avoid any unnecessary lines) and the created shape should be very similar to the initial trees. A sample input file (input.txt with an empty line between the two trees' elements) and a sample output message are provided. Your code will be tested with similar inputs.



```
int main(){
        BT mybt1, mybt2, mybt3;
        mybt1.create(); mybt2.create(); mybt3.create();
        //……… code provided for reading data from input file to fill in mybt1 and mybt2
        mybt3.root = mergeBST(mybt1.root, mybt2.root);
        print(mybt1);
        cout << endl << "****" << endl;
        print(mybt2);
        cout << endl << "****" << endl;
        print(mybt3);
        cout << endl << "****" << endl;


        getchar();
}
```



```
/* The output should be as
follows for the input.txt
3 1 5 4 7
****
5 2 7 1 3 8 0 4
****
8 3 12 1 3 4 15 0 4
****

*/
}
```

Question 2. (30 pts) Assuming that you are given two sorted singly linked list, whose data structures and creation methods are provided, you are asked to write a mergeList function which will create a new sorted linked list from the input lists' elements. mergeList takes 3 arguments (addresses of Linked List structures), where the first 2 are the input linked lists and the 3rd one is the output linked list. Your program shouldn't take any new memory but rather use the nodes of the input lists and link them to each other. After the operation, the original lists' will be empty. The signature of the method and the main function are provided. You are NOT allowed to change the main function and the existing methods. Please only fill in the body part of the mergeList method. Writing extra methods to existing structures is allowed but they could only be invoked by the mergeList function. A sample input file (input.txt with an empty line between the two linked lists' elements) and a sample output message are provided. Your code will be tested with similar inputs. Attention: if the same number appears in both of the input list, it should appear only ONCE in the last list. In such a case, Don't forget to delete the unnecessary (extra) node. The given input file should produce the following output:

```
3 15 18 25 72 88 95 100 150
****
1 3 16 17 20 34
****

****

****
1 3 15 16 17 18 20 25 34 72 88 95 100 150
```

Question 3. (35 pts) Solve the following problem by filling the required places in the skeleton code.
We want to check whether a given input has proper matching parenthesis or not? For example ( ) [ ] and [( )] has proper matching parenthesis but)(or ([ )] not. Let 0,1, 2, 3, 4 denote different types of left parenthesis such as (, [, .. and 5, 6, 7, 8, 9 denote corresponding right parenthesis such as),], ... Therefore 0-5,1-6, 2-7, 3-8, 4-9 are the parenthesis pairs that need to match. For example 1649 or 1496 has proper matching parenthesis but 61 or 1946 not. We will provide you ten digit sequences such as 2334345666 and ask you to output TRUE if it is a proper matching and FALSE if it is not. For example 1111166666, 2740591638 and 1327272786 should return TRUE and 3654112344 should return FALSE. A sample Calico file having many samples and answers is provided from Ninova. Your code will be tested with a similar file.

In this question, you can use any library that you want. Also, you can ignore to check whether the user input contains numerical or alphabetical values.

```
#include <iostream>

#define INPUT_LENGTH 10
using namespace std;

bool check1(char input[INPUT_LENGTH])
{
// YOU WILL IMPLEMENT THIS FUNCTION
//THIS FUNCTION RETURNS TRUE OR FALSE
}
```

```
int main()
{
char input[INPUT_LENGTH];
cout<<"Enter the input:"<<endl;
cin.getline(input,INPUT_LENGTH+1);
if(check(input))
cout<<"TRUE"<<endl;
else
cout<<"FALSE"<<endl;
return 0;
}
```