Military Institute of Science and Technology
B.Sc. in Computer Science and Engineering
Online Examination-1 (Spring) : 20 May, 2021
Subject: CSE 204, Data Structures and Algorithms Sessional-I

Time: 80 minutes **Full Marks: 20**

INSTRUCTIONS:

a. **Question-1** in **Section-A** is **compulsory.**
b. Answer any of the **ONE** question from **Section-B**
c. Figures in the margin indicate **full marks**.

SECTION-A

Q1

During World War II, the Germans used a machine named Enigma to send ciphered (coded) messages for its Military operations in secrecy. Which was then broken by a team of mathematicians in Great Britain that provided the groundwork for the invention of modern computers, the story made famous by the movie "*The Imitation Game*". **10**

Inspired by it, you decided to make a little ciphering machine of your own to send coded messages to your friend. The machine takes the message you want to send to your friend as an input and codes each character of your message to numeric digits as per the equation given below. Note that your machine can send only **8 numeric values at a time.**

Code = (ASCII value of the character + Key) % 20
Where, Key = Last two digits of your ID.

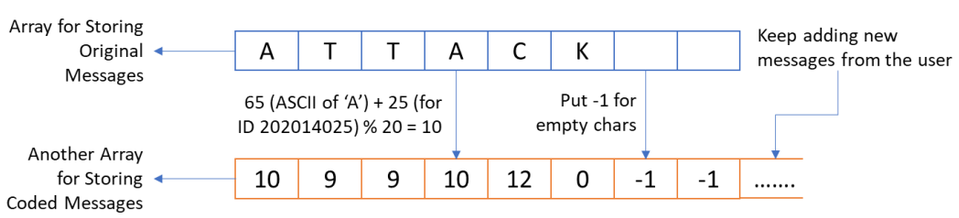
Now, write a C++ code that runs your machine and does the following:

1. Takes input of the message (at most 8 characters) to be sent to your friend and stores it. Only storing the current message is enough.

2. Let the user delete all occurrences of a particular character from the input message.

3. Cipher (turn original input message to coded message) and store it. However, store **ALL** the coded digits in an array for future reference. Make the storage as efficient as possible, that is, **grow the coded message's array** as new ciphers are appended.

   Always add 8 values to the array. Put -1 in remaining spots if there's less than 8 characters.



Note:

1. Use one array to store the original message and one array to continuously store the coded messages. Use Static and Dynamic Array list appropriately.
2. You may use STL Vector to implement DAL.
3. **Code on top of this template. (Click here to download).**

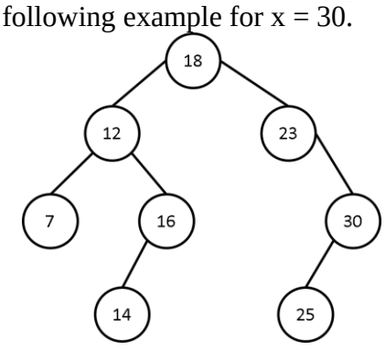
SECTION-B

Q2

This is a very common and straightforward problem of Binary Search Tree. A Binary Search Tree will be given to you and you will have to print all the pairs (a,b) from the Binary Search Tree where: **10**

1. a+b = x; x is a positive integer and given as input
2. a < b

For simplicity you can assume that there are no repeated values in the Binary Search Tree. Check the following example for x = 30.



Here all possible pairs are: {(7, 23), (12, 18), (14,16)}. You can print the pairs in any sequence.

For your convenience a program is written as a template with a function inside the "BST" class named "*void insert(int p)*" that inserts **p** in a Binary Search Tree. Find the template-code *HERE*.

Now your task is to implement the necessary function/s in the "BST" class on the template-code that completes the described scenario and call it when choice=2 is submitted by the user.

Q3

A *double-ended queue* or *deque* (pronounced "deck") is a generalization of a stack and a queue that supports adding and removing items from either the front or the back of the data structure. In the last sessional class, you have implemented a normal queue [also can be found here]. You have to add functions so that this normal *queue* acts like a *deque*. **10**

a. Add *push* function to insert an element at the front.

b. Add *pop* function to remove an element from the rear.

c. Add *complementarySequence* function to find the complement of a DNA sequence using the *deque* data structure. A DNA is a sequence of A, T, C, and G. This function will take DNA sequence and return the complement of that sequence.

d. In the main function, add an option which will take a DNA sequence from the user, call *complementarySequence* function and print the returned result.

Sample Input: ATAACGGA
Sample Output: AGGCAATA