ASSIGNMENT CODES OF SENIORS

Initialize a dynamic array and ask user to give input the initial size and elements of the array. Also create

the following menu:

1. Insert element at the first position of the array

2. Insert element at the last position of the array

3. Insert element at any position of the array

4. Delete an element from the first position of the array

5. Delete an element from the first position of the array

6. Delete an element from any position of the array

7. Update any value of any position of the array

8. Quit

Create different functions for the different options. Make sure, the size of the array is updated after each

insert and delete.

Enter the size of the array: 5

Enter the elements of the array: 10 12 5 70 11

The resultant array: 10 12 5 70 11

Choose one of the option:

1. Insert element at the first position of the array

2. Insert element at the last position of the array

3. Insert element at any middle position of the array

4. Delete an element from the first position of the array

5. Delete an element from the first position of the array

6. Delete an element from any position of the array

7. Update any value of any position of the array

8. Quit

Enter option: 1

Enter value: 4

The resultant array: 4 10 12 5 70 11

Enter option: 2

Enter value: 8

The resultant array: 4 10 12 5 70 11 8

Enter option: 3

Enter position: 2

Enter value: 50

The resultant array: 4 50 10 12 5 70 11 8

Enter option: 4

The resultant array: 50 10 12 5 70 11 8

Enter option: 5

The resultant array: 50 10 12 5 70 11

Enter option:6

Enter position: 8

Entered position is not a valid one.

Enter option:6

Enter position: 4

The resultant array: 50 10 12 70 11

Enter option:7

Enter position: 4

Enter value: 9

The resultant array: 50 10 12 9 11

Enter option:8

ALONG WITH THE CODES YOU NEED TO ADD TWO FUNCTIONS:

1) Implement the following function:

void frequency(int n)

The function will take an integer n and find how many times it has occurred in

that array. For example, if the arraylist contains 1,2,3,3,3,3,4,5 then after calling

frequency(3) the output will be 4.

2) Implement the following function:

void insert\_after\_val (int val, int num\_of\_inputs)

The function will take an integer value val which has to be searched by binary

search and num\_of\_inputs will say how many integers will be added by user in

the arraylist after the value val. Insertion must be maintain a sorted arraylist. For

example, if the arraylist contains 1, 2, 3, 4, 5, 7 then after calling insert\_after\_val

(3,2) the output will be 1,2,3,x1,x2,5,7.

NB: While taking an input to insert in the array list, it will be inserted when it must

be greater than its previous value and less than its next value. Otherwise it will

show an error massage.

For example, if the arraylist contains 1, 2, 3, 4, 5, 7 then after calling

insert\_after\_val (3, 1).

Insert: 8

Output: value can’t be inserted.

3) Implement the following function:

void LeftRotate(int n)

The function will take an integer n and left rotate the array n times. For example,

if the arraylist contains 1,2,3,4,5 then after calling LeftRotate(3) the list will look

like 4,5,1,2,3

4) Implement the following function:

void Reverse(int start, int end)

The function will take two integer start, end and reverse the array from start to

end. For example, if the arraylist contains 1,2,3,4,5,6 then after calling

Reverse(2,4) the list will look like 1,4,3,2,5,6.

5) Implement the following function:

void Max\_Min (int start, int end)

The following function will find the maximum and minimum number in the

arralist. Fpr example, if the arraylist contains 1,9,20,4,8,0 then after calling the

function Max\_Min(0,5) it will give output :

Max: 20

Min: 0

SOLN:

Queue is a linear data structure which follows a particular order in which the operations

are performed. The order may be FIFO (First In First Out) or LILO (Last In Last Out).

Mainly the following four basic operations are performed in the stack:

d. Enqueue: Adds an item in the queue. If the queue is full, then it is said to be an

Overflow condition.

e. Dequeue: Removes an item from the Queue. The oldest item in the queue will be

dequeued. If the queue is empty, then it is said to be an Underflow condition.

f. isEmpty: Returns true if queue is empty, else false.

g. size: Returns the length.

Implement a queue of Integer values. You can define the size of a queue in the code

(#define MAX 5). (NB: Implement it using object oriented approach)

Stack is a linear data structure which follows a particular order in which the operations

are performed. The order may be LIFO (Last In First Out) or FILO (First In Last Out).

Mainly the following four basic operations are performed in the stack:

a. Push: Adds an item in the stack. If the stack is full, then it is said to be an Overflow

condition.

b. Pop: Removes an item from the stack. The items are popped in the reversed order in

which they are pushed. If the stack is empty, then it is said to be an Underflow

condition.

c. Peek or Top: Returns top element of stack.

d. isEmpty: Returns true if stack is empty, else false.

Implement a stack of Integer values. You can define the size of a stack in the code

(#define MAX 10;). (NB: Implement it using object oriented approach)

Home Assignment:

1. Implement a method named merge (Stack s) which will take a stack as an input and

merge it with the corresponding stack. You must maintain the overflow condition

(merge until the corresponding stack is not overflowed).

Example: Let us say, your stack1 is containing 1 2 3 4

Your stack2 is containing 5 6 7 8

After merging stack2 to stack1, the state of stack1 will be 1 2 3 4 8 7 6 5

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.

Question – 1

a. 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”.

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

digits of code at a time.

Code = (ASCII value of the character + Key) % 20

Where, Key = Last two digits of your ID.

Now, write a 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).

10

Page 1 of 2

Question – 2

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:

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.

10

Question – 3

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

Page 2 of 2

Military Institute of Science and Technology

B.Sc. in Computer Science and Engineering

Online Examination-2 (Spring) : 01 July, 2021

Subject: CSE 204, Data Structures and Algorithms Sessional-I

Time: 1 hour 30 minutes Full Marks: 30

Question-1 20

A country consists of some interconnected cities. There are some bidirectional roads

inside a country that connect 2 cities. Length of a bidirectional road is 100 km. Each

country has a capital city. Shortest distance of a city is calculated from the capital city of

that country. Now you will be given a graph consisting of some countries and you have to

find out the terms described in the output section. For simplicity you will assume the

following:

1. The name of a city consists of only lowercase letters except the capital city. The

name of a capital city starts with an uppercase letter and the rest of the letters are

lowercase.

2. The name of a country contains all uppercase letters.

3. String length of a city or country is at least 2.

4. The names of all the cities are different.

5. A country consists of at least 3 cities.

6. If two cities are unreachable from each other then it is guaranteed that they are

from different countries.

Input

First line of input contains a single integer e denoting the total number of bidirectional

roads. Next e lines of input contain 3 space separated strings each: X, Y, Z which denote

that there is a bidirectional road between X and Y and the country name of both X and Y

is Z. If X or Y begins with an uppercase letter then that represents the capital city of Z.

Note that Z is a string that consists of all uppercase letters. Next line of input contains a

single integer q that denotes the number of queries. Next q lines of input contain

a single string P each.

Output

For each query print the following:

1. Name of the country of P [4 marks]

2. Shortest path of P from the capital city of the country of P [10 marks]

3. Shortest distance of P from the capital city of the country of P (in km) [6 marks]

If P is not found in the graph then just print “City not found”. If there are multiple

shortest paths then print any one of them. See the sample output for clarification.

Sample Input Output

Input Output

9

chennai rajasthan INDIA

Dhaka khulna BANGLADESH

khulna rajshahi BANGLADESH

barishal Dhaka BANGLADESH

Delhi chennai INDIA

Delhi rajasthan INDIA

barishal rajshahi BANGLADESH

rajshahi cumilla BANGLADESH

cumilla khulna BANGLADESH

4

cumilla

rajshahi

coxbazar

rajasthan

BANGLADESH

Dhaka -> khulna -> cumilla

200 km

BANGLADESH

Dhaka -> barishal -> rajshahi

200 km

City not found

INDIA

Delhi -> rajasthan

100 km

The following figure is the graph representation of the sample input output. This is

attached for your convenience.

Figure 1

Question - 2 10

Template: Click Here (ideone) or here (drive, .cpp file)

Well, if you did last week’s class, you now know how to implement quicksort to sort an

integer array from smallest to largest number (ascending order).

But, we can do sorting in many different ways and even quicksort can be made more

efficient for specific cases! Now, based on you knowledge from last class, implement the

following -

1) Use quicksort to sort an integer array in descending order. [5 marks]

(Case A of Figure 2.)

2) Modify quicksort so that instead of taking the rightmost element of a partition as

pivot, it takes the minimum number of every partition (including the initial array)

as the pivot. [5 marks]

(Case B of Figure 2.)

3) Modify your entire code so that the quicksort algorithm now works for both

integer numbers and float numbers. You must not write another quicksort

function that uses a float array! [Hint: use templates] [Bonus problem-3

marks]

Note: implement on top of the given template.

Figure 2

Military Institute of Science and Technology

B.Sc. in Computer Science and Engineering

Online-1, Spring 2022

Date: 22 May, 2022

Subject: CSE-204, Data Structures and Algorithms-I Sessional

Time: 60 Minutes Full Marks: 40

ID Name Attendance Marks

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Question-1

Design a singly linked list where each Node contains an integer and a Node-pointer that

points its next node (rightmost Node). The linked list has 4 methods as described in

Table-1(a) considering that the number of elements in the linked list is denoted by n.

Method Description Time

Complexity

void insertAtLast(int v) Inserts v at the tail (end) of the linked list O(1)

void deleteDuplicate(int v) Deletes all the occurrences of v except the first

occurrence from the linked list. If the number of

occurrences of v in the linked list is less than two

then the method deletes nothing.

O(n)

void leftRotate( ) Rotates all the elements of the linked list one

position left from their current position

O(1)

void printReverse( ) Prints all the elements from right to left (reverse

order) separated by a space. No other additional

data structures like array/vector/linked list can be

used. This function can be parameterized if

required.

O(n)

Table-1(a)

Clarification

● Let the current state of the linked list from head to tail is: 2 4 5 7 4 8 9 4 4 9 2.

Now after performing deleteDuplicate(4) all the occurrences of 4 except the first

occurrence will be deleted. So after performing deleteDuplicate(4) the state of the

linked list from head to tail will be: 2 4 5 7 8 9 9 2

● Let the current state of the linked list from head to tail is: 4 5 7 2 1 9. Now after

performing leftRotate( ) all the elements of the linked list will be rotated one position

left from their current position. So after performing leftRotate( ) the state of the linked

list from head to tail will be: 5 7 2 1 9 4

● Let the current state of the linked list from head to tail is: 4 5 7 2 1 9. Now after

performing print( ) the elements of the linked list will be printed as 9 1 2 7 5 4.

Note that any additional data structure for storing the elements of the linked list is

strictly prohibited.

Instructions

Create a menu in your program that takes choice from 1 to 5 where the corresponding method

for each choice is illustrated in Table-1(b).

Choice Method

1 void insertAtLast(int v)

2 void deleteDuplicate(int v)

3 void leftRotate( )

4 void print( )

5 Exit from program

Table-1(b)

Marks Distribution

If the time complexity of the methods is not followed then 50% marks will be reduced.

SL Criteria Allotted Marks Obtained Marks

1 Correct definition of Node 1

2 Correct declaration of the Linked List 2

3 Initializations 1

4 Creating the menu 1

5 Implementation of <void insertAtLast(int v)> 2

6 Implementation of <void deleteDuplicate(int v)> 6

7 Implementation of <void leftRotate( )> 5

8 Implementation of <void printReverse( )> 5

9 Overall correctness 2

Total 25

Question-2

The insertion and deletion sequence of a data structure D is given. Task is to identify the data

structure whether D is a LIFO or FIFO. Note that for a LIFO insertion is named as push,

deletion is named as pop and for a queue insertion is named as enqueue and deletion is named

as dequeue. For simplicity it can be assumed that an underflow or overflow condition never

occurs for the given scenario.

Operations

There are two types of operations: insertion and deletion. insertion is denoted by 1 and

deletion is denoted by 2. For example: (1 x) denotes that x has been inserted in D and (2 y)

denotes that a deletion operation has been performed and the deleted value is y.

Input

First line of input contains a single integer n that denotes the number of operations. The

following n lines indicate n operations where each operation consists of two integers. The

first integer is denoted by t and indicates the type of operation. The second integer is denoted

by v. t = 1 means that v is inserted in D. t = 2 means that a delete operation has been

performed on D and the deleted value is v.

Output

● Print “LIFO” if D is a LIFO

● Print “FIFO” if D is a FIFO

● Print “BOTH” is D follows the characteristics of both LIFO and FIFO

● Print “NONE” if D does not follow the characteristics of either LIFO or FIFO

Sample Test Case

Input Output

5

1 4

1 5

2 4

1 3

2 5

FIFO

6

1 8

1 8

2 8

1 3

2 3

2 8

LIFO

3

1 5

1 5

2 5

BOTH

6

1 4

1 5

2 4

1 3

2 3

2 5

NONE

Clarification

For the first test case, the first input is 5 means that there are a total 5 operations. First

operation is (1 4) means that 4 is inserted in D. In the next operation 5 is inserted. Now in the

third operation, as 4 has been deleted so surely it is not LIFO as 4 has been inserted before 5

and also 4 has been removed earlier than 5. In the forth operation 3 has been inserted. In the

fifth operation 5 has been deleted. Now after the deletion of 4 the oldest element in D is 5. So

surely D it is a FIFO.

For the fourth test case, at first 4 and 5 have been inserted sequentially. Then in the third

operation 4 has been deleted which indicates that D may be a FIFO. Then in the forth

operation 3 has been inserted. In the fifth operation 3 has been deleted but that expresses the

property of a LIFO because 3 is the most recent element of D. So, D is neither a LIFO or

FIFO.

Marks Distribution

SL Criteria Allotted Marks Obtained Marks

1 Correct declaration of appropriate data

structures

2

2 Formulation of appropriate logic 4

3 Identification of “FIFO” 2

4 Identification of “LIFO” 2

5 Identification of “BOTH” 2

6 Identification of “NONE” 2

7 Overall correctness 1

Total 15

Military Institute of Science and Technology

B.Sc. in Computer Science and Engineering

Online-2, Spring 2022

Date: 17 July, 2022

Subject: CSE-204, Data Structures and Algorithms-I Sessional

Time: 105 Minutes Full Marks: 40

ID Name Q-1 Q-2 Total

Question-1

Design a Binary Search Tree B where each node contains a positive integer as key. B has five methods

described in Table-1(a).

Method Description

void insert(int x) Inserts x in B if x > 0 otherwise returns without doing anything

void printByLevel( ) Prints the keys of B from left to right in level wise order. For example,

the following sequence of keys will be generated if the method operates

on the Binary Search Tree illustrated in Figure-1(a)

60 30 90 10 40 62 20 35 45 15 42

void printInorder( ) Prints the keys of B following the inorder traversal (Sorted sequence of

keys)

int maxOfLevel(int k) Returns the maximum key of k-th level in B. The value of

maxOfLevel(k) for different values of k is shown below when the method

operates on the Binary Search Tree shown in in Figure-1(a)

k = 0 → maxOfLevel(0) = 60

k = 1 → maxOfLevel(1) = 90

k = 2 → maxOfLevel(2) = 62

k = 3 → maxOfLevel(3) = 45

k = 4 → maxOfLevel(4) = 42

Note that, root is located at level-0.

void deleteMin( ) Deletes the minimum (leftmost) node of B.

Table-1(a)

Figure-1(a)

Instructions

Create a menu in your program that takes choice from 1 to 6 where the corresponding method for each

choice is mentioned in Table-1(b). Parameters of the methods can be modified in case of recursive

implementation.

Choice Method

1 void insert(int x)

2 void printByLevel( )

3 void printInorder( )

4 int maxOfLevel(int k)

5 void deleteMin( )

6 Exit from program

Table-1(b)

Marks Distribution

SL Criteria Allotted Marks Obtained Marks

1 Correct definition of BST 2

2 Implementation of <void insert(int x)> 4

3 Implementation of <void printByLevel( )> 6

4 Implementation of <void printInorder( )> 4

5 Implementation of <int maxOfLevel(int k)> 8

6 Implementation of <void deleteMin( )> 6

Total 30

Question-2

Consider that you are given an sorted array of integers P and a single integer X. Write a recursive

function named binarySearch that returns the number of integers in P that are greater than X. Note

that, if the number of elements in P is denoted by n then the time complexity of the function named

binarySearch must be O(log2n). Any definition of binarySearch that is not recursive and the time

complexity is more than O(log2n) will be rejected!!!

Input

First line of input is n denoting the number of integers in P. Next line contains n number of space

separated integers (sorted in ascending order) denoting the elements of P. Next line contains X.

Output

Print a single integer that is returned from the function named binarySearch.

Sample Input Output

Input Output

10

15 18 20 20 20 20 20 22 22 28

20

3

12

15 18 20 20 20 20 22 22 26 26 33 34

25

4