


National University of Computer and Emerging Sciences, Lahore Campus

	Course Name:	Data Structures	Course Code:	CS2001
	Degree Program:	BS (CS, SE, DS)	Semester:	Fall 2022
	Exam Duration:	60 Minutes	Total Marks:	20
	Paper Date:	28-Sept-2022	Weight	15
	Section:	ALL	Page(s):	6
	Exam Type:	Midterm-I		

Student : Name. _____ Roll _____ Section: 1

Instruction/Notes: Attempt all questions. Answer in the space provided. You can ask for rough sheets but will not be attached with this exam. **Answers written on rough sheet will not be marked.** Do not use pencil or red ink to answer the questions. In case of confusion or ambiguity make a reasonable assumption.

Question1:

(Marks: 10)

Your task is to write a C++ function "deleteSubSequence" that removes a desired subsequence from a singly linked list of integers that store binary digits such that each node either stores zero or one. This function must delete all the sub lists / sequences containing binary representation that are positive power of 2 ($2^0=1$ is not included). For Example, $2^1 = 10$, $2^2 = 100$ so on. Below is a table that contains sample inputs and outputs.

Input:	1->1->0->0->1->0->1	1->0->0->0->1->1->0	0->1->1->1	1->0
Output:	1->1	1	0->1->1->1	null

$2^1 = 10$
 $2^2 = 100$
 $2^3 = 1000$
 $2^4 = 10000$

Assume that the singly linked list has dummy/sentinel head and tail nodes. Traverse the list using an iterator (BDS-3A and BDS-3B can do it without iterator) and remove the required subsequences. Write down the time complexity of your function. If you need any helper function, write down their definition as well. Note that this function is a member function of class list and less efficient implementations will get lesser reward.

~~void deleteSubSequence (Iterator obj) .~~

~~Node* temp = obj->head;~~

~~List<int> :: Iterator obj (head->next);~~

~~for (obj; obj != end(); obj++)~~

~~if (*obj Node* temp2 =~~

~~Iterator obj2 = obj->curr->next~~

~~if (~~

Done on next page

Question2:**(Marks: 5)**

Compute the time complexity of the function func1. First write the time complexity expression and then compute the big-oh of the time complexity function. Compute the tight bounds

```

void func2(int arr[], int l, int m, int r){
    int i, j, k;
    - int n1 = m - l + 1; ①
    - int n2 = r - m; ② → ①
    - int *L = new int[n1], *R = new int[n2];
      ① ②+1 ③
    for (i = 0; i < n1; i++)
        L[i] = arr[l + i]; ④
    for (j = 0; j < n2; j++)
        R[j] = arr[m + 1 + j];

    i = 0;    j = 0;    k = l;
    while (i < n1 && j < n2){
        if (L[i] <= R[j])
            arr[k++] = L[i++];
        else
            arr[k++] = R[j++];
    }
    while (i < n1)
        arr[k++] = L[i++];

    while (j < n2)
        arr[k++] = R[j++];

    delete []L;
    delete []R;
}

```

```

void func1(int arr[], int n){
    int curr_size; 1
    int left_start; 1
    for (curr_size=1; curr_size<=n-1; curr_size = 2*curr_size)
    {
        1
        for (left_start=0; left_start<n-1; left_start += 2*curr_size)
        {
            ⑤ int mid = min(left_start + curr_size - 1, n-1);
            //assume it returns the min of two numbers
            ⑥ int right_end = min(left_start + 2*curr_size - 1, n-1);
            ⑦ func2(arr, left_start, mid, right_end);
        }
    }
}

```

1 2 2


Question3:

(Marks: 5)

We have an implementation of unsorted doubly LinkedList. Suppose it has its implementation with head pointer (i.e., pointers to the first node of linked list) only. Which of the following can be implemented in constant time? Justify your answer.

- a) Insertion at the end of LinkedList
- b) Deletion of the last node of LinkedList

National University of Computer and Emerging Sciences, Lahore Campus

	Course Name:	Data Structures	Course Code:	CS2001
	Degree Program:	BS (CS, SE, DS)	Semester:	Fall 2022
	Exam Duration:	60 Minutes	Total Marks:	20
	Paper Date:	12-Nov-2022	Weight	15
	Section:	ALL	Page(s):	5
	Exam Type:	Midterm-II		

Student Name: _____ Roll No. _____ Section _____

Instruction/Notes: Attempt all questions. Answer in the space provided. You can ask for rough sheets but will not be attached with this exam. **Answers written on rough sheet will not be marked.** Do not use pencil or red ink to answer the questions. In case of confusion or ambiguity make a reasonable assumption.

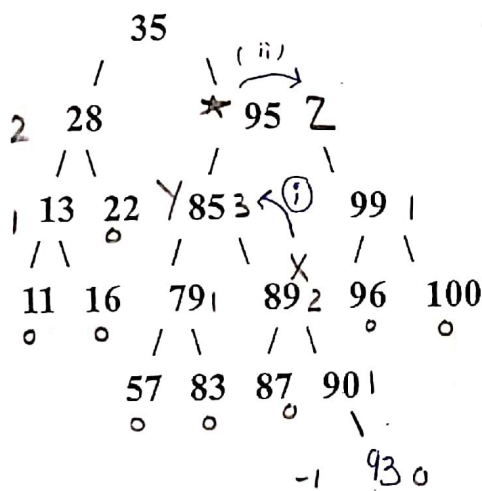
$$8 + 7 = 15$$

Question 1: [CLO: 1, 3]

(Marks: 5+5)

- a) Insert the value 93 in the following AVL tree and redraw the tree after rebalancing. You must show all working with the names of imbalance cases, nodes, and rotations performed.

X Y Z

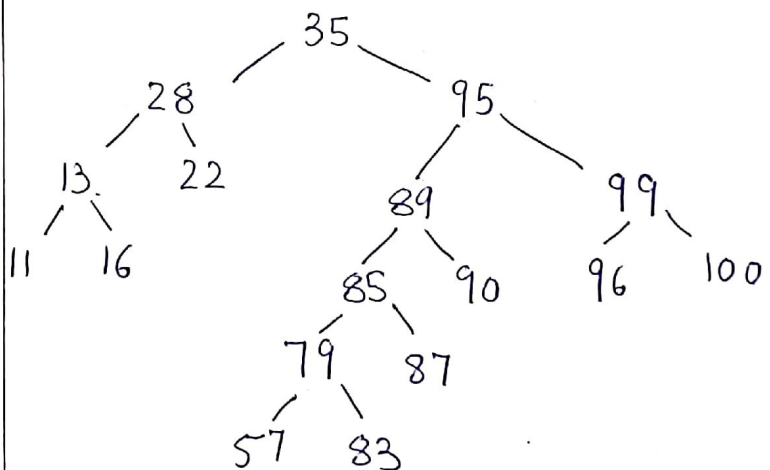


Z = 95
Y = 85
X = 89

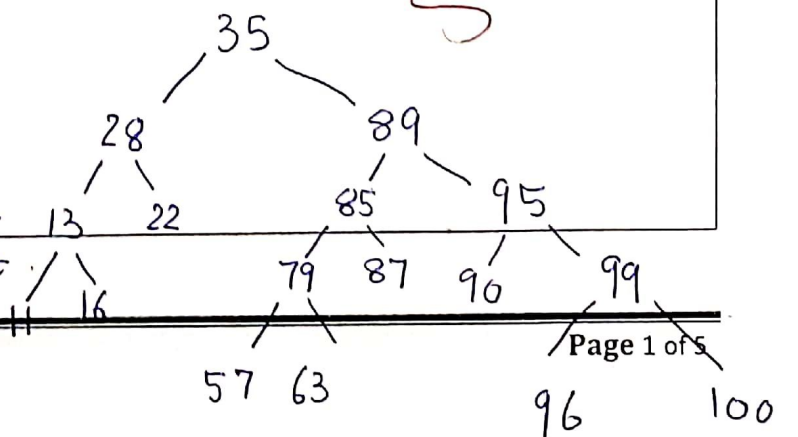
Unbalanced node 95

left right case

left rotate Y = 85



right rotate Z



Imbalance Node:	95
Imbalance Case:	left right case
Rotations Performed:	1 rotate Y left

ii) rotate Z right

b) For each of the scenarios given below, suggest the most appropriate data structure chosen from the list and give appropriate reason of your choice. LIFO
(Arrays, linked-list (single, double, circular), Queue, Stack, tree(BST, AVL))

To create an index of important terms in a an electronic book	BST tree . 0.5
To check whether a given seat number is reserved in a cinema or not	doubly linked list X
To implement play list in repeat mode of a music player	circular linked list . ✓
To implement undo and redo functionality in a text editor	Stack 0.5
To implement token system at bank.	Queue . ✓

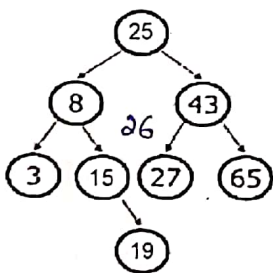
Question 2: [CLO: 3]

(Marks: 10)

Write a **RECURSIVE** C++ function **FINDVALUE** in an Integer-based AVL tree class that takes a positive integer value as input and find the node with the smallest integer larger than or equal to the given input value. The function returns the node's data value and it must not take more than $O(\lg n)$ time.

EXAMPLE

AVL- Tree



Input is 8, then the value returned is 8

As 8 is the smallest integer larger than or equal to the given value 8

Input is 26, then the value returned is 27


As the integer 26 does not exists and 27 is the smallest integer larger than 26 in the given AVL tree

Input is 22, then the value returned is 25

As 22 does not exists and 25 is the smallest integer larger than 22

Input is 67, then the value returned is -1

As there is no integer equal to or larger than 67 in the given tree AVL tree

	Course:	Data Structures	Course Code:	CS218
	Program:	BS(Computer Science)	Semester:	Fall 2022
	Duration:	10 Minutes	Total Marks:	10
	Paper Date:	20-Oct-22	Weight	
	Section:	D	Page(s):	1
	Exam:	Quiz 3(a)	Roll No:	
			Section:	D

10

What is the functionality of following recursive function? Give your answer in 2 lines only also compute its answer for n=27

```

int fun1(int n)
{
    if(n == 1)
        return 0;
    else
        return 1 + fun1(n/2);
}
    
```

for of random n = 40

n = 40

n = 105

fun1.
 1 + (13)
 1 + 1 + fun1.
 1 + 1 + 1 + fun(3)
 1 + 1 + 1 + 1 + fun(1)

1 + fun1(20)
 1 + 1 + fun1(10)
 1 + 1 + 1 + fun1(5)
 1 + 1 + 1 + 1 + fun(2)
 1 + 1 + 1 + 1 + 1 + fun(1)
 answer = 5


1 + 52
 1 + 1 + 26
 1 + 1 + 1 + 13
 1 + 1 + 1 + 3
 answer = 6

if input is 27 answer will be 4

it is computing $\log_2 n$ it gives the answer in whole number of $\log_2 n$

550
 1 + 275
 1 + 137
 1 + 1 + 68
 1 + 1 + 1 + 34
 17

National University of Computer and Emerging Sciences, Lahore Campus


	Course:	Data Structures	Course Code:	CS218
	Program:	BS(Computer Science)	Semester:	Fall 2022
	Duration:	10 Minutes	Total Marks:	10
	Paper Date:	22-Sept-22	Weight	
	Section:	D	Page(s):	1
	Exam:	Quiz 2(a)	Roll No:	
			Section:	D

Write a function reverse() in C++ in the class of singly linked list that reverses the direction of all the elements in the linked without use of iterators. Assume that there are dummy head and tail nodes.

Before function call
temp L
Head -> 1-> 3 -> 5-> 7 -> 12 -> 19 ->tail

After function call
temp
Head -> 19-> 12 -> 7-> 5 -> 3 -> 1 ->tail

National University of Computer and Emerging Sciences, Lahore Campus

	Course:	Data Structures	Course Code:	CS218
	Program:	BS(Computer Science)	Semester:	Fall 2022
	Duration:	10 Minutes	Total Marks:	10
	Paper Date:	8-Sept-22	Weight	
	Section:	D	Page(s):	1
	Exam:	Quiz 1(b)	Roll No:	
			Section:	

Compute the time complexity of the following code fragment using step count method (frequency of each instruction) in the worst case. Also compute the big-oh of the time complexity function.

```
int sum,i,j,n;  
sum = 0; → 1)  
cin>>n; → 1)  
for (i=1;i*i<=n;i=i+1)  
    for (j=1;j*j<n;j=j*2)  
        sum++;
```


National University of Computer and Emerging Sciences, Lahore Campus



Course:	DS Lab	Course Code:	CL 2001
Program:	BS (Computer Science)	Semester:	Spring 2022
Duration:	2 Hours	Total Marks:	50
Paper Date:	2 nd Jan, 2023	Weight:	45%
Section:	BCS-3A To BCS-3G	Page(s):	3
Exam:	Lab Final term	Reg. No.	

Read the below Instructions Carefully:

- Understanding the question statement is also part of the exam, so do not ask for any clarification. In case of any ambiguity, make suitable assumptions.
- You have to complete the exam in 2 hrs. No extra time will be given for submission.
- Submit a single .cpp file for each question named as **21L-1122 (Q#)**
- Place all .cpp files into the **folder** named as **21L-1122**
- Submit folder on **cactus** by following path: \\cactus1\ Xeon\ Fall 2022\ Mamoona Akbar\ DS\ Final\sec
- Your code should be **intended** and **commented** properly. Use **meaningful variable names**.
- It is your responsibility to save your code from being copied. All matching codes will be considered cheating cases. **PLAGIARISM** will result in forwarding of case to **Disciplinary Committee** and **negative marks** in Midterm.

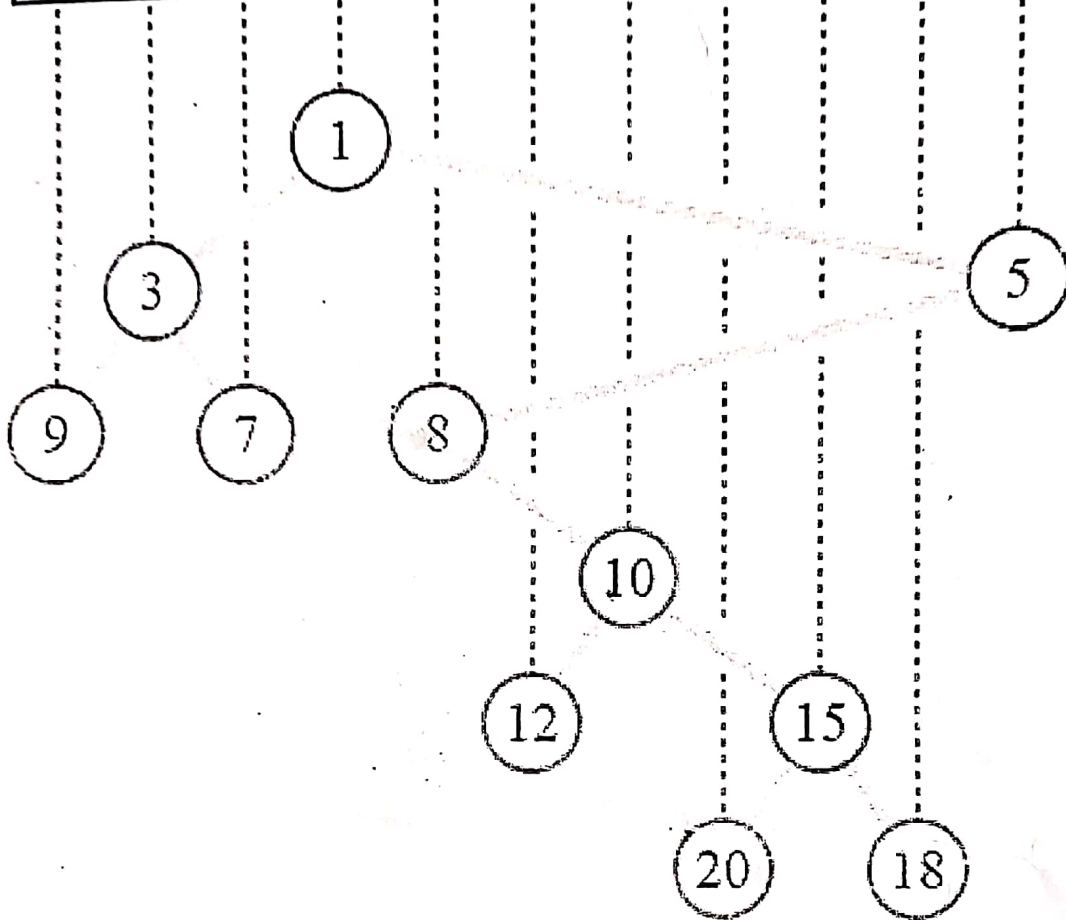
Question 1: Marks 25

Write an efficient algorithm to construct a **Cartesian tree** from an array using **inorder traversal**. A Cartesian tree is a **binary tree** with the **heap property**: the parent of any node has a smaller value than the node itself.

For example, the following figure shows an example of a Cartesian tree derived from the sequence of numbers in inorder:



9	3	7	1	8	12	10	20	15	18	5
---	---	---	---	---	----	----	----	----	----	---



9 7 3
18 15
8

1 3 5

1 3 5 7 8 9

min 1.

3
arr[j]

3 9 7

while size != 0
startindex++

Question 2: Marks 25

Given an $M \times N$ matrix of characters, find all occurrences of a given string in the matrix. We are allowed to search the string in all eight possible directions, i.e., North, West, South, East, North-East, North-West, South-East, South-West. Note that there should not be any cycles in the output path.

For example, consider the following matrix of characters,

0 1 2 3 4

0	'D'	'E'	'M'	'X'	'B'
1	'A'	'O'	'E'	'P'	'E'
2	'D'	'D'	'C'	'Q'	'D'
	'E'	'B'	'E'	'D'	'S'
	'C'	'P'	'Y'	'E'	'N'

0 1 2
0
1
2
3
4

If the given string is “CODE”, following are all its occurrences in the matrix:

C(2, 2) O(1, 1) D(0, 0) E(0, 1)
C(2, 2) O(1, 1) D(2, 0) E(3, 0)
C(2, 2) O(1, 1) D(2, 1) E(1, 2)
C(2, 2) O(1, 1) D(2, 1) E(3, 0)
C(2, 2) O(1, 1) D(2, 1) E(3, 2)
C(2, 2) O(2, 3) D(2, 4) E(1, 4)
C(2, 2) O(2, 3) D(3, 3) E(3, 2)
C(2, 2) O(2, 3) D(3, 3) E(4, 3)

Note: used DFS to solve this problem.

National University of Computer and Emerging Sciences, Lahore Campus



Course: Data Structures Lab
Program: BS(Computer Science)
Duration: 1:30 hours
Paper Date: 24th Oct, 2022
Section: BCS-3D
Exam: Midterm

Course Code: CL218
Semester: Fall 2022
Total Marks: 100
Weight: 25%
Page(s): 2
Roll No:

Instruction/Notes:

- We will check your code for plagiarism. If plagiarism is found, it will result in F grade in lab.
- In case of any ambiguity make suitable assumption.
- No cell phones are allowed. Sharing of USBs or any other items is **not allowed**.
- You are not allowed to have any helping code with you.
- Submission path is \\cactus1\Xeon\Fall 2022\Fareeha Ashfaq\DS 3D\Mid\ D1/D2 (Submit your code in your respective section).
- Make different .cpp files for both questions named as roll#_Q# (21L-1234_Q1)
- Make folder with their roll# (21L-1234) and places all files (both .cpp + .txt file)
- Don't submit the .zip folder.

Question 1: (60 marks)

A CPU can execute many processes. However, at a given time, a CPU can execute the instructions of only one process (not true for modern CPUs, nonetheless). To manage the execution of n processes, we make use of what is known as "CPU Scheduler". We will write a basic type of scheduler which will work as follows: Suppose we have 10 processes named p_1, p_2, \dots, p_{10} . Each process has some number of instructions $n_1, n_2, n_3, \dots, n_{10}$, respectively. Now to execute all the processes, we first execute some instructions, let's say 3 instructions of process p_1 , and then we will execute 3 instructions of p_2 , so on and so forth. After that we will restart from process p_1 and execute 3 instructions of p_1 , then p_2 , then p_3 , so on and so forth. Then the cycle begins again. So, the processes are being executed by CPU in circular fashion. If a process has finished its instructions during this cycle, then we remove that process from this cycle. The given example below depicts the scenario:

Suppose we have 3 processes:

Process_id: p1

Total_Instructions: 7

Process_id: p2

Total_Instructions 6

Process_id: p3

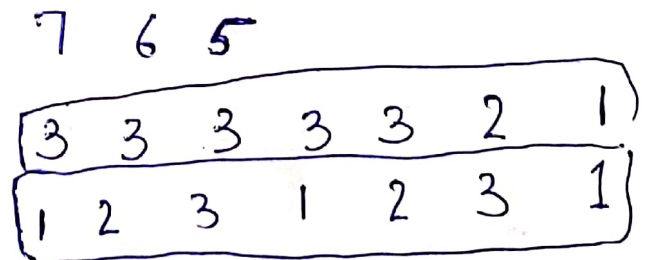
Total_Instructions 5

Scheduler Output:

3 instructions of p1 executed.

3 instructions of p2 executed.

3 instructions of p3 executed.



3 instructions of p1 executed.
 3 instructions of p2 executed.
 p2 has finished execution.
 2 instructions of P3 executed.
 p3 has finished execution.
 1 instruction of p1 executed.
 p1 has finished execution.

Implement your scheduler function using the queue. The function will take a file name as argument. The file contains information about all the processes to be executed. A sample file is also given. The first number in the file tells about the number of instructions that the CPU will execute, of a process p, at one time. The second number tells us about the total number of processes in the file. Test your function in main with the given file.

```
void scheduler(string processFile);
```

Question 2: (40 marks)

You are given the head of a singly linked-list. The list can be represented as:

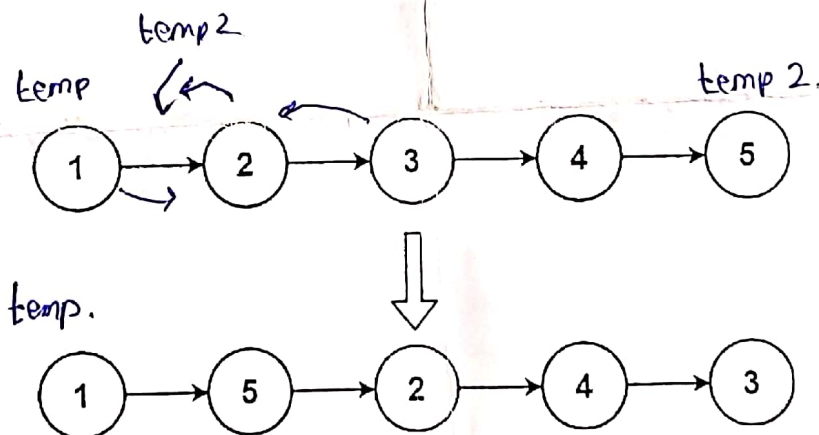
$L_0 \rightarrow L_1 \rightarrow \dots \rightarrow L_{n-1} \rightarrow L_n$

Reorder the list to be on the following form:

$L_0 \rightarrow L_n \rightarrow L_1 \rightarrow L_{n-1} \rightarrow L_2 \rightarrow L_{n-2} \rightarrow \dots$

You may not modify the values in the list's nodes. Only nodes themselves may be changed.

Example:



Input: head = [1, 2, 3, 4, 5]

Output: [1, 5, 2, 4, 3]

5/2

1 2 3 4 5 6 7

1 7 2 3 4 5 6

1 2 3 4

1 4 2 3

1 7 2 6 3 4 5 1 5 2 3 4