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: [redacted] **Roll No.** [redacted] **Section:** [redacted]

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.

12

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

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 (SList *obj)
{
    SList <int> :: Iterator iter;
    int position_start = 0, position_end = 0;
    int open = 0;
    for (iter = obj.begin(); iter != obj.end(); ++iter)
    {
        if (obj -> data == 1)
        {   open = 1;
            position_start ++;
            position_end ++;
            continue;
        }
        if (obj -> data == 0 && open == 1)
        {
            if (obj -> next -> data == 1)
            {   for (int i = position_start; i <= position_end; i++)
                {   Delete AtPosition(i);
            else
            {   position_end ++;
```

5

```
Iterator start ()
{   Iterator obj;
    obj = head->next;
    return obj;
}
Iterator end ()
{   Iterator obj;
    obj = tail->prev;
    return obj;
}
Iterator operator ++ ( )
{   Iterator obj;
    obj.*curr = curr->next;
    return obj;
}
void Delete At Position (int val)
{   Node * temp ;
    Node * temp 1;
    Node * temp 2;
    while ( ; temp2 ! ->next ! = NULL; )
    {   temp 1 = curr -> prev;
        temp 2 = temp 1;
        temp 1 = temp2 ->next;
    }
    temp 1 = temp 2 -> next;
    temp 2 = temp 1;
```

Malfunction

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;
        int n2 = r - m;

        int *L= new int[n1], *R=new int[n2];

        for (i = 0; i < n1; i++)
                L[i] = arr[l + i];
        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;
int left_start;
for (curr_size=1; curr_size<=n-1; curr_size = 2*curr_size)
{
        for (left_start=0; left_start<n-1; left_start += 2*curr_size)
        {
                int mid = min(left_start + curr_size - 1, n-1);
              //assume it returns the min of two numbers
                int right_end = min(left_start + 2*curr_size - 1, n-1);
                func2(arr, left_start, mid, right_end);
        }
}
}
```

Answer:

$T(n)$ of func2 $= 1+1+1+n1+1+n2+1+1+n1+n2+n1+1+n2+1+1+1$

$= 3n1 + 3n2 + 10$

$T(n)$ of func1 $= \log_2 \log_2 (\log_2 (curr\_size))$

$T(n) = 3 + \log_2 (\log_2 (curr\_size)) + \log_2 (\log_2 (curr\_size)) \cdot (3n1 + 3n2 + 10)$

$O(n) = \log_2 (\log_2 (\log_2 (curr\_size)))$

CamScanner

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

a) It cannot be done in constant time as it has ~~prev~~ no tail pointer which can append where it is pointed to the newly inserted node at the end of the linked list.

b) This cannot also be done in constant time as the ~~previous pointer~~ tail node cannot be deleted and the tail's previous node cannot point to null. There is ~~no~~ need to iterate through the list.

5

CamScanner