

18CSC201J
DS LAB EXERCISES
UNIT-2

DATE: 11.12.2021

NAME: ANINDYA SHANKAR DASGUPTA
REG NO: RA2011026010120
CSE Q1 (AI-ML)

RA2011026010120

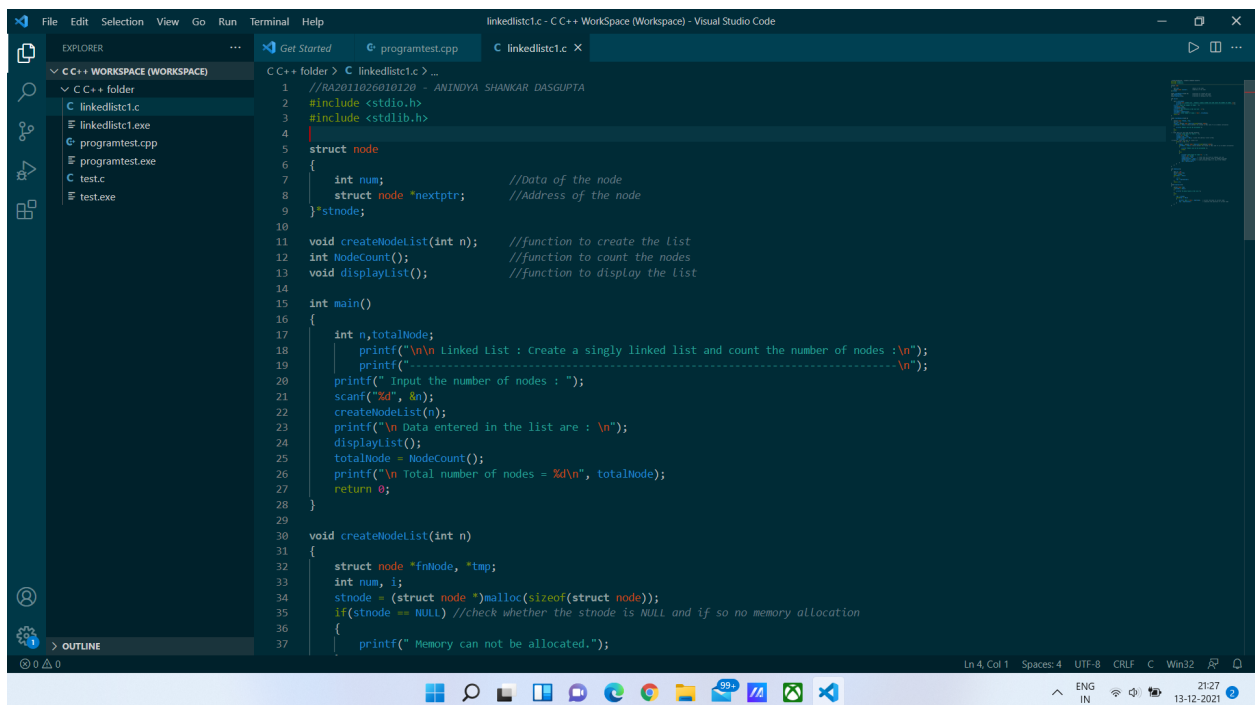
Simple Problems:

Q:

1. (2.) Write a program in C to create a singly linked list of n nodes and count the number of nodes

A:

- Approach:
 1. There are certain ambiguities in the questions, like if the list to be created is empty or has some nodes already. We must run some conditions before inserting to check if the list is empty or not.
 2. If the list is empty, and we create a linked list of 'n' nodes then the number of nodes automatically will be 'n' itself, and there is no need to write a code separately for that purpose.
- Code (with documentation):



```
1 //RA2011026010120 - ANINDYA SHANKAR DASGUPTA
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 struct node
6 {
7     int num; //Data of the node
8     struct node *nextptr; //Address of the node
9 }*stnode;
10
11 void createNodeList(int n); //function to create the list
12 int NodeCount(); //function to count the nodes
13 void displayList(); //function to display the list
14
15 int main()
16 {
17     int n,totalNode;
18     printf("\n\n Linked List : Create a singly linked list and count the number of nodes :\n\n");
19     printf("-----\n\n");
20     printf(" Input the number of nodes : ");
21     scanf("%d", &n);
22     createNodeList(n);
23     printf("\n Data entered in the list are : \n");
24     displayList();
25     totalNode = NodeCount();
26     printf("\n Total number of nodes = %d\n", totalNode);
27     return 0;
28 }
29
30 void createNodeList(int n)
31 {
32     struct node *frNode, *tmp;
33     int num, i;
34     stnode = (struct node *)malloc(sizeof(struct node));
35     if(stnode == NULL) //check whether the stnode is NULL and if so no memory allocation
36     {
37         printf(" Memory can not be allocated.");
```

```
File Edit Selection View Go Run Terminal Help linkedlist1.c - C C++ Workspace (Workspace) - Visual Studio Code
EXPLORER
C C++ WORKSPACE (WORKSPACE)
  C C++ folder
    linkedlist1.c
    linkedlist1.exe
    programtest.cpp
    programtest.exe
    test.c
    test.exe
C C++ folder > C linkedlist1.c > ...
37 // printf("%d", *num); can not be allocated.
38 }
39 else
40 {
41 // reads data for the node through keyboard
42 printf(" Input data for node 1 : ");
43 scanf("%d", &num);
44 stnode->num = num;
45 stnode->nextptr = NULL; //Links the address field to NULL
46 tmp = stnode;
47 //Creates n nodes and adds to linked list
48 for(i=2; i<=n; i++)
49 {
50     fnNode = (struct node *)malloc(sizeof(struct node));
51     if(fnNode == NULL) //check whether the fnNode is NULL and if so no memory allocation
52     {
53         printf(" Memory can not be allocated.");
54         break;
55     }
56     else
57     {
58         printf(" Input data for node %d : ", i);
59         scanf("%d", &num);
60         fnNode->num = num; // Links the num field of fnNode with num
61         fnNode->nextptr = NULL; // Links the address field of fnNode with NULL
62         tmp->nextptr = fnNode; // Links previous node i.e. tmp to the fnNode
63         tmp = tmp->nextptr;
64     }
65 }
66 }
67 }
68
69 int NodeCount()
70 {
71     int ctr = 0;
72     struct node *tmp;
73     tmp = stnode;
74     while(tmp != NULL)
```

```
File Edit Selection View Go Run Terminal Help linkedlist1.c - C C++ Workspace (Workspace) - Visual Studio Code
EXPLORER
C C++ WORKSPACE (WORKSPACE)
  C C++ folder
    linkedlist1.c
    linkedlist1.exe
    programtest.cpp
    programtest.exe
    test.c
    test.exe
C C++ folder > C linkedlist1.c > ...
74     while(tmp != NULL)
75     {
76         ctr++;
77         tmp = tmp->nextptr;
78     }
79     return ctr;
80 }
81 void displayList()
82 {
83     struct node *tmp;
84     if(stnode == NULL)
85     {
86         printf(" No data found in the list.");
87     }
88     else
89     {
90         tmp = stnode;
91         while(tmp != NULL)
92         {
93             printf(" Data = %d\n", tmp->num); // prints the data of current node
94             tmp = tmp->nextptr; // advances the position of current node
95         }
96     }
97 }
```

- Dry-run:

For sample input/output shown below:

Block-1	Declared struct node (node of list of struct type) Accordingly, declared node variables to be used: stnode , fnode , tmp They are dynamically allocated during runtime.
Block-2	n = user-input number of nodes = 12
Block-3 Function: createNodelist	stnode: head node ; fnode: created new node ; tmp: previous node in list stnode->data = 8; stnode->next = NULL; tmp = stnode i=2: fnode->data = 4; fnode->next=NULL; newly created node tmp->next=fnode; so, tmp : previous node in list is linked to newly created node. tmp = tmp->next; newly linked node becomes previous node for next node to be inserted. i=3: fnode->data = 9; same procedure followed . . i=12: fnode->data = 16; fnode->NULL; tmp->next = fnode; tmp = tmp->next; so, tmp is now last node, has data = 16 and points to NULL Linked list is now: 8->4->9->13->0->44->6->125->2->-5->22->16
Block-4 Function to print list	tmp: used as current node checked if tmp = NULL ,i.e if list is empty List non-empty tmp = stnode : first node/head Then, tmp is traversed through the list till it reaches end (12 times) Each time, current node value is printed: tmp = head: data = 8 is printed and next node is passed to tmp : tmp = tmp->next tmp = node2: data = 4 tmp = node12: data = 16 is printed. tmp -> next then points to NULL (last node) So, loop is broken
Block-5	Using the same traversal method as display function, A counter variable is incremented till tmp (current node used for traversal) reaches end of linked list, to find out number of nodes. Then it is also displayed. Here counter-variable = ctr = 12 is printed.

- Input/Output:

Input-

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Anindya\Documents\VSCode\C++ folder> cd "c:\Users\Anindya\Documents\VSCode\C++ folder\" ; if ($?) { gcc linkedlist1.c -o linkedlist1 } ; if ($?) { .\linkedlist1 }

Linked List : Create a singly linked list and count the number of nodes :
-----
Input the number of nodes : 12
Input data for node 1 : 8
Input data for node 2 : 4
Input data for node 3 : 9
Input data for node 4 : 13
Input data for node 5 : 0
Input data for node 6 : 44
Input data for node 7 : 6
Input data for node 8 : 125
Input data for node 9 : 2
Input data for node 10 : -5
Input data for node 11 : 22
Input data for node 12 : 16
```

Output-

```
Data entered in the list are :
Data = 8
Data = 4
Data = 9
Data = 13
Data = 0
Data = 44
Data = 6
Data = 125
Data = 2
Data = -5
Data = 22
Data = 16
```

```
Total number of nodes = 12
```

```
PS C:\Users\Anindya\Documents\VSCode\C++ folder> █
```

- Result:

1. Time-complexity:

For non-looped statements, each is considered to take up 1 unit (as they are executed only once).

Let sum of these be = cnst (constant value)

There are 3 loops:

for loop: 1 to n : n times

while loop: n times

while loop: n times

Total time-complexity = cnst + 3*n

Asymptotic upper-bound time-complexity = $O(\text{cnst} + 3*n) = O(n)$

2. Space-complexity:

Each node = 8 bytes.

n nodes = 8*n bytes.

1 extra node variable = 8 bytes (fnnode)

int n,ctr,num,i = 4*4 = 16 bytes

Total space complexity = 16+8+8*n bytes = 24+8*n bytes

Asymptotic upper-bound space-complexity = $O(24 + 8*n) = O(n)$

RA2011026010120

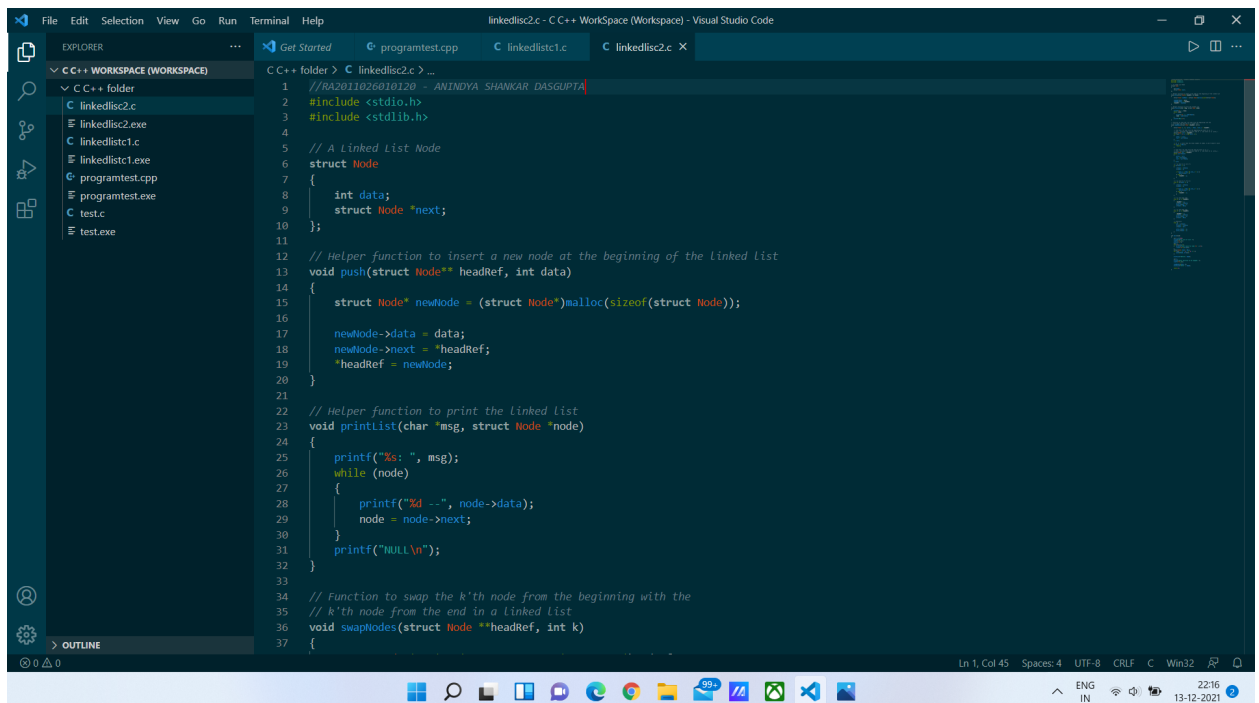
Scenario based or Difficult Problems:

Q:

1. (16.) Given a singly linked list, swap kth node from beginning with kth node from end. Swapping of data is not allowed, only pointers should be changed.

A:

- Approach:
 1. Not given if the list is empty or not. We must check explicitly.
 2. Value of k entered may exceed the size of the list. We should print an error message for that scenario.
 3. The list is supposed to be given but it is not given in question and we have to take user-input.
 4. Value of k entered may correspond to the exact middle node of the list, in which case the same list is printed as output without changes.
- Code (with documentation):



```
1 //RA2011026010120 - ANINDYA SHANKAR DASGUPTA
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 // A Linked List Node
6 struct Node
7 {
8     int data;
9     struct Node *next;
10 };
11
12 // Helper function to insert a new node at the beginning of the Linked List
13 void push(struct Node** headRef, int data)
14 {
15     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
16     newNode->data = data;
17     newNode->next = *headRef;
18     *headRef = newNode;
19 }
20
21 // Helper function to print the Linked List
22 void printList(char *msg, struct Node *node)
23 {
24     printf("%s: ", msg);
25     while (node)
26     {
27         printf("%d --", node->data);
28         node = node->next;
29     }
30     printf("NULL\n");
31 }
32
33 // Function to swap the k'th node from the beginning with the
34 // k'th node from the end in a Linked List
35 void swapNodes(struct Node **headRef, int k)
36 {
37 }
```

```
37 struct Node *x, *y, *prev_x = NULL, *prev_y = *headRef;
38
39 // Find the k'th node from the beginning and store it in 'x'.
40 // Also, calculate the previous node of 'x' and store it in 'prev_x'.
41 struct Node* curr = *headRef;
42 for (int i = 1; i < k && curr; i++)
43 {
44     prev_x = curr;
45     curr = curr->next;
46 }
47 x = curr;
48
49 // If 'k' is more than the total number of nodes, x and y doesn't exist
50 if (curr == NULL) {
51     return;
52 }
53
54 // Find the k'th node from the end and store it in 'y'.
55 // Also, calculate the previous node of 'y' and store it in 'prev_y'.
56 struct Node* ptr = *headRef;
57 while (curr->next)
58 {
59     prev_y = ptr;
60     ptr = ptr->next;
61     curr = curr->next;
62 }
63 y = ptr;
64
65 // Y is next to X (X -> Y)
66 if (x->next == y)
67 {
68     x->next = y->next;
69     y->next = x;
70 }
71
72 if (prev_x != NULL && prev_x != x) {
73     prev_x->next = y;
74 } else {
```

```
74     } else {
75         *headRef = y;
76     }
77 }
78
79 // X is next to Y (Y -> X)
80 else if (y->next == x)
81 {
82     y->next = x->next;
83     x->next = y;
84
85     if (prev_y != NULL && prev_y != y) {
86         prev_y->next = x;
87     } else {
88         *headRef = x;
89     }
90 }
91
92 // X is the head node
93 else if (x == *headRef)
94 {
95     *headRef = y;
96     y->next = x->next;
97     prev_y->next = x;
98     x->next = NULL;
99 }
100
101 // Y is the head node
102 else if (y == *headRef)
103 {
104     *headRef = x;
105     x->next = y->next;
106     prev_x->next = y;
107     y->next = NULL;
108 }
109
110 // otherwise
```



```

111     else {
112         ptr = y->next;
113         y->next = x->next;
114         x->next = ptr;
115     }
116     prev_x->next = y;
117     prev_y->next = x;
118 }
119
120
121 int main(void)
122 {
123     int n, arr[50];
124     printf("Enter size of list: ");
125     scanf("%d", &n);
126     printf("\n");
127     int i;
128     for(i=0; i<n; i++)
129     { printf("Enter value for node %d: ", i+1);
130       scanf("%d", &arr[i]);
131     }
132     struct Node* head = NULL;
133     for (int i = n - 1; i >= 0; i--) {
134         push(&head, arr[i]);
135     }
136
137     printList("Before", head);
138
139     int k;
140     printf("Enter position to be swapped: ");
141     scanf("%d", &k);
142
143     swapNodes(&head, k);
144     printList("After ", head);
145
146     return 0;
147 }

```

- Dry-run:

Block-1	Initializing nodes and other variables newNode; headref; node
Block-2 Inserting in list	First, size of list = n = 5 Insertion as follows: i=-; list: empty : (headref->NULL) i=0: list: headref->1->NULL i=1: list: headref->2->NULL . . i=4: list: headref->1->2->3->4->5->NULL
Block-3 Swapping	k = 2 traversing using curr variable from first node to kth node curr->data = 2; curr->next = 3 (node-3) Store this value in x: x = 2 Similarly, traverse from last node to kth last node curr->data = 4 Store this value in y: y=4 Check if adjacent nodes Then swap x with y then pass these values to the previously kth nodes Now list: headref->1->4->3->2->5->NULL
Block-4	Printing new list

- Input/Output:

Input-

```
Enter size of list: 5
Enter value for node 1: 1
Enter value for node 2: 2
Enter value for node 3: 3
Enter value for node 4: 4
Enter value for node 5: 5
```

Output-

```
Before: 1 --2 --3 --4 --5 --NULL
Enter position to be swapped: 2
After : 1 --4 --3 --2 --5 --NULL
PS C:\Users\Anindya\Documents\VSCode\C++ folder> █
```

- Result:

1. Time complexity:

For each statement complexity = 1

Let there be cnst no. of such statements: complexity = cnst

Loops :

for loop : n times (enter array values)

for loop : n times (push array values to linked list)

while loop : n times (traversing through list to print) x2

while loop: k times (finding kth element) x2

Total time-complexity = $\text{cnst} + 2*n + 2*n + 2*k$

Asymptotic upper-bound time-complexity = $O(\text{cnst} + 4*n + 2*k) = O(n)$

2. Space complexity:

No. of nodes = n : Each node = 8 bytes : $8*n$ bytes

2 extra nodes for traversing : 16 bytes

2 extra nodes for swapping : 16 bytes

int n,k,arr[50] : $4*3 = 12$ bytes

Total space-complexity = $8*n + 44$

Asymptotic upper-bound space-complexity = $O(8*n + 44) = O(n)$

18CSC201J
DS LAB EXERCISES
UNIT-3

DATE: 12.12.2021

NAME: ANINDYA SHANKAR DASGUPTA
REG NO: RA2011026010120
CSE Q1 (AI-ML)

RA2011026010120

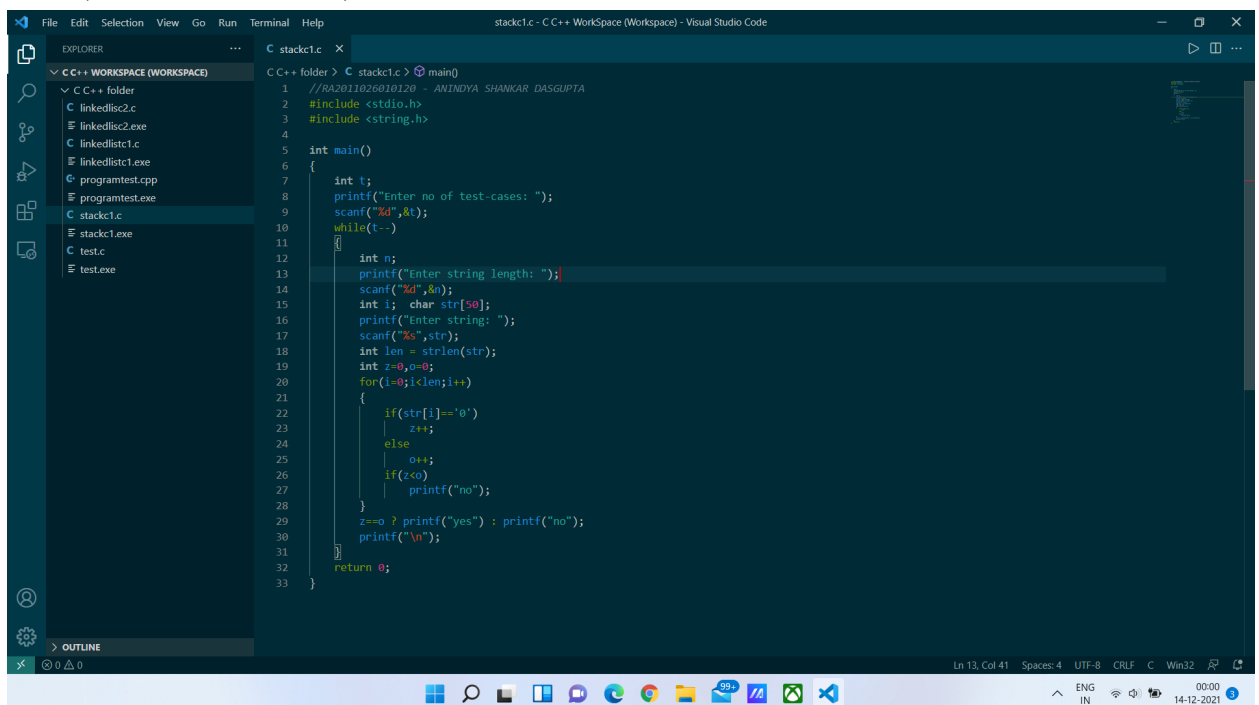
Simple Problems:

Q:

1. (1.) Given binary string **str** of size **N** the task is to check if the given string is valid or not. A string is called valid if the number of **0**'s equals the number of **1**'s and at any moment starting from the left of the string number **0**'s must be greater than or equals to the number of **1**'s.

A:

- Approach:
 1. Size of string is not required for string input in C. So, n is an unused value in this program. We may omit this.
 2. If some other input other than 0 or 1 is given, it will not be valid. We must print an error message for such a case.
 3. Other than that, the program should be fine.
- Code (with documentation):



```
1 //RA2011026010120 - ANINDYA SHANKAR DASGUPTA
2 #include <stdio.h>
3 #include <string.h>
4
5 int main()
6 {
7     int t;
8     printf("Enter no of test-cases: ");
9     scanf("%d",&t);
10    while(t-->0)
11    {
12        int n;
13        printf("Enter string length: ");
14        scanf("%d",&n);
15        int i; char str[50];
16        printf("Enter string: ");
17        scanf("%s",str);
18        int len = strlen(str);
19        int z=0,o=0;
20        for(i=0;i<len;i++)
21        {
22            if(str[i]!='0')
23                z++;
24            else
25                o++;
26            if(z<o)
27                printf("no");
28        }
29        z==o ? printf("yes") : printf("no");
30        printf("\n");
31    }
32    return 0;
33 }
```

- Dry-run:

Block-1	Initialization to garbage values
Block-2	Input no of test cases t = 2 For 2 test cases <ol style="list-style-type: none"> 1. size = 4 : str = 0011 len = 4 i=0: z = 1 i=1: z = 2 i=2: o = 1 i=3: o = 4 z < o : false for all i z == o (after loop) : true : print “yes” 2. Similarly, for 2nd test case: size = 3 : str = 001 But after loop z!=o : print “no”

- Input/Output:

Input-

```
PS C:\Users\Anindya\Documents\VSCode\C++ folder> cd "c:\Users\Anindya\Documents\VSCode\C++ folder\" ; if ($?) { gcc stackc1.c -o stackc1 } ; if ($?) { .\stackc1 }
Enter no of test-cases: 2
Enter string length: 4
Enter string: 0011
yes
Enter string length: 3
Enter string: 001
no
PS C:\Users\Anindya\Documents\VSCode\C++ folder> █
```

Output-

```
PS C:\Users\Anindya\Documents\VSCode\C++ folder> cd "c:\Users\Anindya\Documents\VSCode\C++ folder\" ; if ($?) { gcc stackc1.c -o stackc1 } ; if ($?) { .\stackc1 }
Enter no of test-cases: 2
Enter string length: 4
Enter string: 0011
yes
Enter string length: 3
Enter string: 001
no
PS C:\Users\Anindya\Documents\VSCode\C++ folder> █
```

- Result:

1. Time complexity:

Non-loop statements : 4

while loop : t times

for loop : n times : nested in while : $n \cdot t$ times

Total time-complexity = $4 + t \cdot n$

Asymptotic upper-bound time complexity = $O(4 + t \cdot n) = O(t \cdot n) = O(n)$

2. Space complexity:

string = character array of size n : $1 \cdot n$ bytes = n bytes

int i, len, z, o : $4 \cdot 4$ bytes = 16 bytes

Total space complexity = $n + 16$ bytes

Asymptotic upper-bound space complexity = $O(n + 16) = O(n)$

RA2011026010120

Scenario based or Difficult Problems:

Q:

1. (10.) Given an integer array **arr** of size **N** and two integers **K** and **M**, the task is to find **M** largest sums of **K** sized subarrays.

A:

- Approach:
 1. Value of K may exceed array size N. We must print an error for that.
 2. N must be greater than 1. Otherwise the program is invalid as the single element will be the answer itself.
 3. M may exceed the possible number of sums based on N and M.

- Code (with documentation):

```
1 //RA2011020010120 - ANINDYA SHANKAR DASGUPTA
2 #include <stdio.h>
3
4 int main()
5 {
6     int t;
7     printf("Enter no of test cases: ");
8     scanf("%d",&t);
9     while(t-->0)
10     {
11         int n,m,k;
12         printf("Enter array size, subarray size and no of largest sums: ");
13         scanf("%d %d %d",&n,&m,&k);
14         int i,a[100],b[50],j,l,s;
15         printf("Enter array: ");
16         for(i=0;i<n;i++) //reading array from user
17             scanf("%d",&a[i]);
18         i=0; l=0;
19         while(i<=n-k)
20         {
21             s=0;
22             for(j=i;j<=i+k;j++) //storing sum of each such subarray as elements of b
23                 s+=a[j];
24             b[l]=s;
25             l++;
26             i++;
27         }
28         int p = l,temp;
29         for(i=0;i<p-1;i++) //bubble-sort to sort b array in descending order
30             for(j=0;j<p-1-i;j++)
31             {
32                 if(b[j]<b[j+1])
33                 {
34                     temp=b[j];
35                     b[j]=b[j+1];
36                     b[j+1]=temp;
37                 }
38             }
39         printf("Required output : ");
40         for(i=0;i<m;i++)
41             printf("%d ",b[i]);
42         printf("\n");
43     }
44     return 0;
45 }
```

```
31 {
32     if(b[j]<b[j+1])
33     {
34         temp=b[j];
35         b[j]=b[j+1];
36         b[j+1]=temp;
37     }
38 }
39 printf("Required output : ");
40 for(i=0;i<m;i++)
41     printf("%d ",b[i]);
42 printf("\n");
43 }
44 return 0;
45 }
```


- Dry-run:

Block - 1	Initialize t t = 2 For each test-case t we input n,m,k
Block - 2	t=2 n=5,k=2,m=3 Array: a[5]: {10,11,10,11,12} Then we compute all possible sums in subarrays of k = 3 units: i=0;l=0; s = 21: b[0]=21 : i=1 : l=1 s = 21: b[1]=21 : i=2 : l=2 s = 21: b[2]=21 : i=3 : l=3 s = 21: b[3]=23 : i=4 : l=4 b[]={21,21,21,23} We use bubble-sort to sort b[] in descending order b[]={23,21,21,21} Now we just print the first 3 elements of b[] Output: 23 21 21 Same process for test-case 2 (t=1)

- Input/Output:

Input-

```
PS C:\Users\Anindya\Documents\VSCode\C++ folder> cd "c:\Users\Anindya\Documents\VSCode\C++ folder\" ; if ($?) { gcc stackc2.c -o stackc2 } ; if ($?) { .\stackc2 }
Enter no of test cases: 2
Enter array size, subarray size and no of largest sums: 5
2
3
Enter array: 10
11
10
11
12
Required output : 23 21 21
Enter array size, subarray size and no of largest sums: 5
5
1
Enter array: 1
2
3
4
5
Required output : 15
PS C:\Users\Anindya\Documents\VSCode\C++ folder> █
```

Output-

```
PS C:\Users\Anindya\Documents\VSCode\C++ folder> cd "c:\Users\Anindya\Documents\VSCode\C++ folder\" ; if ($?) { gcc stackc2.c -o stackc2 } ; if ($?) { .\stackc2 }
Enter no of test cases: 2
Enter array size, subarray size and no of largest sums: 5
2
3
Enter array: 10
11
10
11
12
Required output : 23 21 21
Enter array size, subarray size and no of largest sums: 5
5
1
Enter array: 1
2
3
4
5
Required output : 15
PS C:\Users\Anindya\Documents\VSCode\C++ folder> █
```

- Result:

1. Time complexity:

Non-loop statements : 4

while-loop: $12 \cdot t$: no of test cases

for-loop: $n : t \cdot n$: input array

while-loop : $t \cdot (n-k+1)$

for-loop : $t \cdot (n-k+1) \cdot k$

for-loop: $t \cdot (n-k+1)$

for-loop: $t \cdot (n-k+1) \cdot \log(n-k+1)$

For-loop: $t \cdot m$

Total time-complexity = $4 + t \cdot (12 + m + (n-k+1)(1 + k + 1 + \log(n-k+1)))$

Asymptotic upper-bound time-complexity = $O(n \log n)$

2. Space complexity:

int a[], b[] : $4 \cdot (n + (n-k+1))$ bytes

int i, j, l, s, temp, p : $4 \cdot 6 = 24$ bytes

Total space-complexity = $24 + 4 \cdot (2 \cdot n + k - 1)$ bytes

Asymptotic upper-bound space complexity = $O(n)$
