18CSC201J

DS LAB EXERCISES

UNIT-3

DATE: 12.12.2021

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

RA2011026010120

Simple Problems:

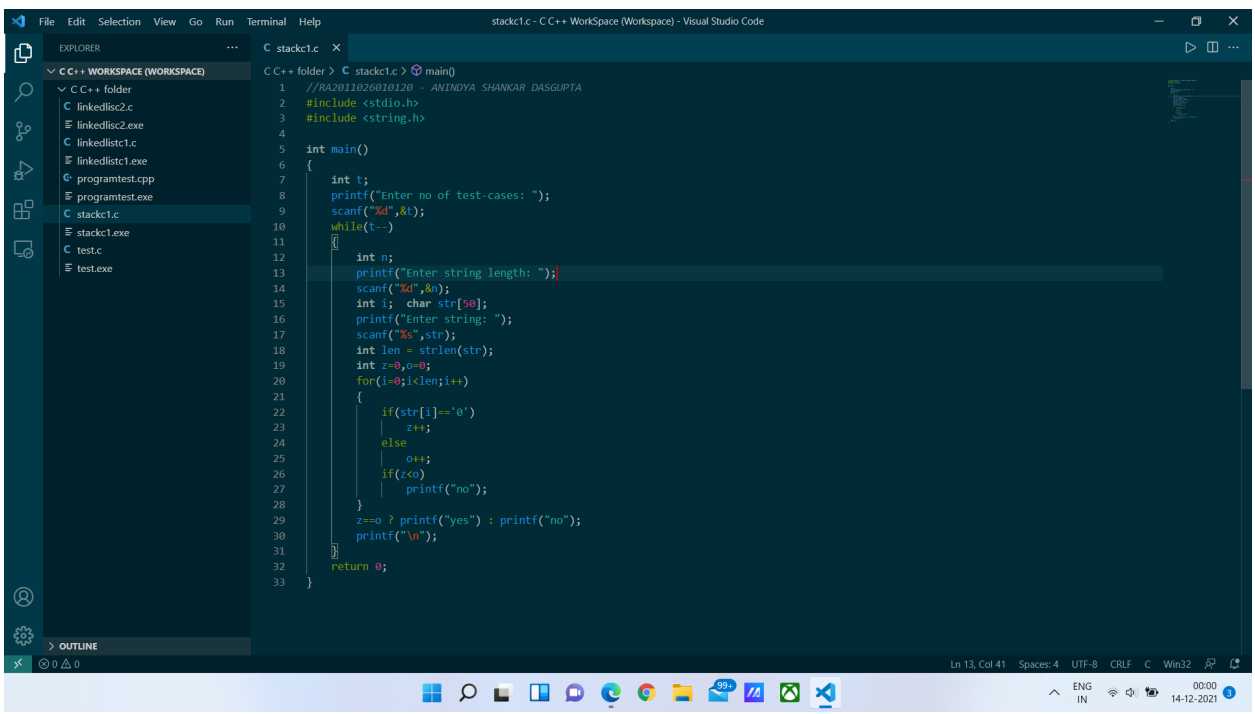
Q:

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):



- <u>Dry-run:</u>

Block-1	Initialization to garbage values
Block-2	Input no of test cases t = 2 For 2 test cases 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”

- <u>Input/Output:</u>

Input-

```
PS C:\Users\Anindya\Documents\VSCode\C C++ folder> cd "c:\Users\Anindya\Documents\VSCode\C 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 C++ folder>
```

Output-

```
PS C:\Users\Anindya\Documents\VSCode\C C++ folder> cd "c:\Users\Anindya\Documents\VSCode\C 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 C++ folder>
```

- Result:

    1. Time complexity:
       Non-loop statements : 4
       while loop : t times
       for loop : n times : nested in while : n*t times
       Total time-complexity = 4 + t*n
       Asymptotic upper-bound time complexity = O(4+t*n) = O(t*n) = O(n)

    2. Space complexity:
       string = character array of size n : 1*n bytes = n bytes
       int i,len,z,o : 4*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.

- <u>Code (with documentation):</u>

```c
//RA2011026010120 - ANINDYA SHANKAR DASGUPTA
#include <stdio.h>

int main()
{
    int t;
    printf("Enter no of test cases: ");
    scanf("%d",&t);
    while(t--)
    {
        int n,m,k;
        printf("Enter array size, subarray size and no of largest sums: ");
        scanf("%d %d %d",&n,&k,&m);
        int i,a[100],b[50],j,l,s;
        printf("Enter array: ");
        for(i=0;i<n;i++)                                    //reading array from user
            scanf("%d",&a[i]);
        i=0;    l=0;
        while(i<=n-k)
        {
            s=0;
            for(j=i;j<i+k;j++)                              //storing sum of each such subarray as elements of b
                s=s+a[j];
            b[l]=s;
            l++;
            i++;
        }
        int p = i,temp;
        for(i=0;i<p-1;i++)
            for(j=0;j<p-i-1;j++)                            //bubble-sort to sort b array in descending order
            {
                if(b[j]<=b[j+1])
                {
                    temp=b[j];
                    b[j]=b[j+1];
                    b[j+1]=temp;
                }
```

```c
            }
        printf("Required output : ");
        for(i=0;i<m;i++)
            printf("%d ",b[i]);
        printf("\n");
    }
    return 0;
}
```

- <u>Dry-run:</u>

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)

- <u>Input/Output:</u>

Input-

```
PS C:\Users\Anindya\Documents\VSCode\C C++ folder> cd "c:\Users\Anindya\Documents\VSCode\C 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 C++ folder>
```

Output-

```
PS C:\Users\Anindya\Documents\VSCode\C C++ folder> cd "c:\Users\Anindya\Documents\VSCode\C 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 C++ folder>
```

- <u>Result:</u>
  1. <u>Time complexity:</u>
     Non-loop statements : 4
     while-loop: 12*t : no of test cases
     for-loop: n : t*n : input array
     while-loop : t*(n-k+1)
     for-loop : t*(n-k+1)*k
     for-loop: t*(n-k+1)
     for-loop: t*(n-k+1)*log(n-k+1)
     For-loop: t*m
     Total time-complexity = 4+t*(12+m+(n-k+1)(1+k+1+log(n-k+1)))
     Asymptotic upper-bound time-complexity = O(nlogn)

  2. <u>Space complexity:</u>
     int a[],b[] : 4*(n+(n-k+1)) bytes
     int i,j,l,s,temp,p : 4*6 = 24 bytes
     Total space-complexity = 24+4*(2*n+k-1) bytes
     Asymptotic upper-bound space complexity = O(n)