SESSION 1:

Problem 1:

DP - Sliding Window Sum

ID:10528 Solved By 818 Users

The program must accept **N** integers and an integer **K** representing the sliding window size as the input. The program must print the sum of integers in each sliding window of size K among the N integers as the output.

Boundary Condition(s):
1 <= K <= N <= 10^6
0 <= Each integer value <= 10^4

Input Format:
The first line contains N.
The second line contains N integer values separated by a space.
The third line contains K.

Output Format:
The first line contains the sum of integers in each sliding window of size K among the N integers.

Example Input/Output 1:
Input:
7
10 5 90 20 5 30 15
3

Output:
105 115 115 55 50

Explanation:
Here, the sliding window size K = 3.
The sum of integers in the first sliding window is 105 (10 + 5 + 90).
The sum of integers in the second sliding window is 115 (5 + 90 + 20).
The sum of integers in the third sliding window is 115 (90 + 20 + 5).
The sum of integers in the fourth sliding window is 55 (20 + 5 + 30).
The sum of integers in the fifth sliding window is 50 (5 + 30 + 15).

Example Input/Output 2:
Input:
8
56 90 50 84 80 56 48 58
2

Output:
146 140 134 164 136 104 106

Max Execution Time Limit: 90 millisecs

Reconnect

Sliding Window Sum

N = 11 k = 4

1 2 3 2 4 5 1 5 8 9 10

Output:

8 11 14 12 15 19 23 32

Code:

```
#include <stdio.h>

int main()
{
    int n,k;
    scanf("%d", &n);
    int arr[n];
    int i=0;

    for(;i<n;i++){
        scanf("%d",&arr[i]);
    }
    scanf("%d", &k);
    int sum=0;

    for( i=0;i<k;i++)
        sum=sum+arr[i];
    printf("%d ",sum);

    for(i=1;i<n-k+1;i++){
        sum=sum-arr[i-1]+arr[i+k-1];
        printf("%d ",sum);
    }
    return 0;
}
```

```
#include<stdio.h>
#include<stdlib.h>
7
10 5 90 20 5 30 15
3
int main()
{
    int N,K;
    scanf("%d",&N);
    int arr[N];
    for(int index=0; index < N; index++){
        scanf("%d",&arr[index]);
    }
    scanf("%d",&K);
    int sum = 0;
    for(int index=0; index < K; index++){
        sum += arr[index];
    }
    printf("%d ",sum);
    for(int index=1; index <= N-K; index++){
        sum = sum - arr[index-1];
        sum = sum + arr[index+K-1];
```

```c
    }
    scanf("%d",&K);
    int sum = 0;
    for(int index=0; index < K; index++){
        sum += arr[index];
    }
    printf("%d ",sum);
    for(int index=1; index <= N-K; index++){
        sum = sum - arr[index-1];
        sum = sum + arr[index+K-1];
        printf("%d ",sum);
    }
    return 0;
}
```

Problem 2:

DP - Vowels in Sliding Window

ID:10529 Solved By 789 Users

The program must accept an integer **K** and a string **S** containing only alphabets as the input. The program must print the count of vowels in each sliding window of size K in the string S as the output.

Boundary Condition(s):
1 <= K <= Length of S <= 10^5

Input Format:
The first line contains K.
The second line contains S.

Output Format:
The first line contains the count of vowels in each sliding window of size K in the string S.

Example Input/Output 1:
Input:
3
environment

Output:
1 1 1 2 1 1 1 1 1

Explanation:
Here, the sliding window size K = 3.
The count of vowels in the 1st sliding window is 1 (**e**nv).
The count of vowels in the 2nd sliding window is 1 (n**v**i).
The count of vowels in the 3rd sliding window is 1 (v**i**r).
The count of vowels in the 4th sliding window is 2 (**i**ro).
The count of vowels in the 5th sliding window is 1 (r**o**n).
The count of vowels in the 6th sliding window is 1 (**o**nm).
The count of vowels in the 7th sliding window is 1 (n**m**e).
The count of vowels in the 8th sliding window is 1 (m**e**n).
The count of vowels in the 9th sliding window is 1 (**e**nt).

Example Input/Output 2:
Input:
2
SKILLRACK

Output:
0 1 1 0 0 1 1 0

Max Execution Time Limit: 10 millisecs

```c
#include<stdio.h>
#include<stdlib.h>

int isVowel(char ch){
    ch = tolower(ch);
    return ch=='a' || ch == 'e' || ch=='i' || ch == 'o' || ch == 'u';
}
```

```c

int main()
{
    int K;
    char str[100];
    scanf("%d\n%s",&K,str);
    int N = strlen(str), vowelCount=0;
    for(int index=0; index < K; index++){
        if(isVowel(str[index])){
            vowelCount++;
        }
    }
    printf("%d ",vowelCount);
    for(int index=1; index <= N-K; index++){
        if(isVowel(str[index-1])){
            vowelCount--;
        }
        if(isVowel(str[index+K-1])){
            vowelCount++;
        }
        printf("%d ",vowelCount);
    }
    return 0;
}

```

```c

int isVowel(char c)
{
    c=tolower(c);
    if(c=='a' || c=='e' || c=='i' || c=='o' || c=='u')
    return 1;
    return 0;
}
int main()
{
    int k,v=0;
    scanf("%d",&k);
    char s[100000];
    scanf("%s",s);
    for(int i=0;i<k;i++)
    if(isVowel(s[i]))
    v++;
    printf("%d ",v);
    for(int i=1;i<=strlen(s)-k;i++)
    {
        if(isVowel(s[i+k-1]))
        v++;
        if(isVowel(s[i-1]))
        v--;
        printf("%d ",v);
    }
}
```

Code:

```c
#include <stdio.h>
#include <string.h>
int isVowel(char ch){
    ch=tolower(ch);
    return ch=='a' ||ch=='e' ||ch=='i' ||ch=='o' ||ch=='u';
}
int main()
{
    int k;
    char str[100];
    scanf("%d\n%s",&k,str);
    int n=strlen(str),vowelcount=0;
```

```
    for(int index=0;index<k;index++){
        if(isVowel(str[index])){
            vowelcount++;
        }
    }
    printf("%d ",vowelcount);
    for(int index=1;index<=n-k;index++){
        if(isVowel(str[index-1])){
            vowelcount--;
        }
        if(isVowel(str[index+k-1])){
            vowelcount++;
        }
        printf("%d ",vowelcount);
    }
    return 0;
}
```

SESSION 2:

LACS-Elite-S002 (DP-S002) TCE-2023 28-Dec-21 (FN)

Solved Challenges 0/2

Boy Chocolate or Ice-cream

ID:10530 Solved By 869 Users

A boy can have either a chocolate(**C**) or an ice-cream(**I**) on a given day. But to avoid catching cold, his mom has prevented him from having ice-cream on consecutive days. The program must print the number of ways W in which the boy can have chocolate or ice-cream over the period of **N** days.

Boundary Condition(s):
1 <= N <= 80

Input Format:
The first line contains N.

Output Format:
The first line contains W.

Example Input/Output 1:
Input:
3

Output:
5

Explanation:
The 5 ways to have over three days are
C C C
C C I
C I C
I C C
I C I

Example Input/Output 2:
Input:
5

Output:
13

Max Execution Time Limit: 500 millisecs

Boy Chocolate or Ice-cream - 10530

```java
import java.util.*;
public class Hello {

    public static void main(String[] args) {
                //Your Code Here
                Scanner sc=new Scanner(System.in);
                int n=sc.nextInt();
        int c[]=new int[n];
        int ic[]=new int[n];
        c[0]=1;
        ic[0]=1;
        int i;
        int tot[]=new int[n];
        for(i=1;i<n;i++)
        {
            c[i]=c[i-1]+ic[i-1];
            ic[i]=c[i-1];
            tot[i]=c[i]+ic[i];
        }
        System.out.println(tot[i-1]);
        }
}
```

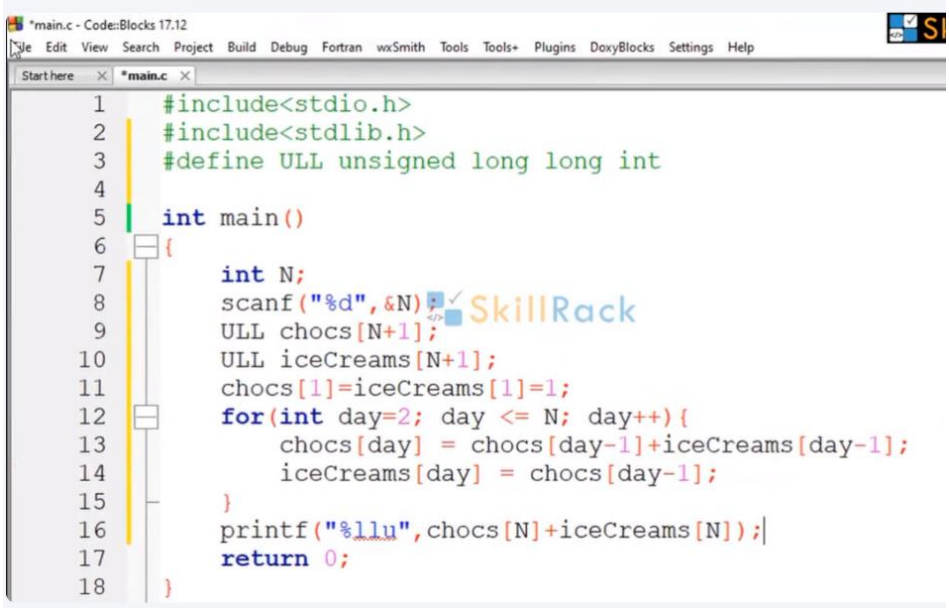
Code:

Boy Chocolate or Ice-cream - 10530

```java
import java.util.*;
public class Hello {

    public static void main(String[] args) {
                //Your Code Here
                Scanner scanObj = new Scanner(System.in);
                long n = scanObj.nextInt();
                long choc=1, ice=1; long total=0;

                for(int i=2; i<=n; i++)
                {
                    total = choc+ice;
                    ice = choc;
                    choc = total;
                }
                System.out.println(ice+choc);

        }
}
```

Code:

```java
import java.util.*;
public class Main
{
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
int n=sc.nextInt();
int c[]=new int[n];
int ic[]=new int[n];
c[0]=1;
ic[0]=1;
int i;
int tot[] = new int[n];
for(i=1;i<n;i++){
    c[i]=c[i-1]+ic[i-1];
    ic[i]=c[i-1];
    tot[i]=c[i]+ic[i];
}
System.out.println(tot[i-1]);
}
}
```



Problem 2:

Boy Ice-cream Kth day

ID:10531 Solved By 846 Users

A boy can have either a chocolate(C) or an ice-cream(I) on a given day. But to avoid catching cold, his mom has prevented him from having ice-cream on consecutive days. As the boy was adamant, his mom gave a relaxation that on every Kth day, the boy can have ice-cream even if he ate ice-cream the previous day. The program must print the number of ways W in which the boy can have chocolate or ice-cream over the period of N days.

Boundary Condition(s):
1 <= N <= 50
2 <= K <= 100

Input Format:
The first line contains N and K separated by a space.

Output Format:
The first line contains W.

Example Input/Output 1:
Input:
3 2

Output:
6

Explanation:
The 6 ways to have over three days are
C C C
C C I
C I C
I C C
I C I
I I C (as K=2, on the second day ice-cream can be had even on successive days)

Max Execution Time Limit: 500 millisecs

```
#include<stdio.h>
#include<stdlib.h>
#define ULL unsigned long long int
```

```c
int main()
{
    int N, K;
    scanf("%d%d",&N,&K);
    ULL chocs[N+1];
    ULL iceCreams[N+1];
    chocs[1]=iceCreams[1]=1;
    for(int day=2; day <= N; day++){
        chocs[day] = chocs[day-1]+iceCreams[day-1];
        iceCreams[day] = chocs[day-1];
        if(day%K == 0){
            iceCreams[day] += iceCreams[day-1];
        }
    }
    printf("%llu",chocs[N]+iceCreams[N]);
    return 0;
}
```

Code:

```java
import java.util.*;
public class Main
{
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
int n=sc.nextInt();
int k=sc.nextInt();
int c[]=new int[n];
int ic[]=new int[n];
c[0]=1;
ic[0]=1;
int i;
int tot[] = new int[n];
for(i=1;i<n;i++){
    c[i]=c[i-1]+ic[i-1];
    ic[i]=c[i-1];
    if(i%k==0){
        ic[i]+=ic[i-1];
    }
}
System.out.println(c[n-1]+ic[n-1]);
    }
}
```

SESSION 3:

LACS-Elite-S003 (DP-S003) TCE-2023 28-Dec-21 (AN)

Solved Challenges 0

DP - Array Elements Equal Left & Right Sum

ID:10610 Solved By 647 Users

The program must accept **N** integers and print the integers where the sum of all integers present to it's left is equal to the sum of all integers present to it's right.
Note: At least one such integer is always present in the given integers.

Boundary Condition(s):
3 <= N <= 10^6
-10^6 <= Each integer value <= 10^6

Input Format:
The first line contains N.
The second line contains N integers separated by a space.

Output Format:
The first line contains the integer value(s) separated by a space.

Example Input/Output 1:
Input:
6
7 2 1 -5 5 5

Output:
-5 5

Example Input/Output 2:
Input:
4
20 10 50 30

Output:
50

Max Execution Time Limit: 80 millisecs

Code:

```
import java.util.*;
public class Main
{
public static void main(String[] args) {
    Scanner in=new Scanner(System.in);
    int n=in.nextInt();
    int[] arr= new int[n];
    int[] sum=new int[n];
    for(int i=0;i<n;i++){
        arr[i]=in.nextInt();
    }

    int sum1=0;
    for(int i=0;i<n;i++){
```

```
        sum1=arr[i]+sum1;
        sum[i]=sum1;
    }
    int lsum,rsum;
    for(int i=0;i<n;i++){
        lsum=sum[i]-arr[i];
        rsum=sum[n-1]-sum[i];
        if(lsum==rsum){
            System.out.print(arr[i]+" ");
        }
    }
}
}
```

Reconnect

```
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int N = sc.nextInt();
    int arr[] = new int[N+1];
    long sumArr[] = new long[N+1];
    for(int index=1; index <= N; index++){
        arr[index] = sc.nextInt();
        sumArr[index] = sumArr[index-1]+arr[index];
    }
    long totalSum = sumArr[N];
    for(int index=1; index <= N; index++){
        if(sumArr[index]-arr[index] == totalSum-sumArr[index]){
            System.out.print(arr[index]+" ");
        }
    }

}
```

LACS-Elite-S003 (DP-S003) TCE-2023 28-Dec-21 (AN)

Solved Challenge

DP - Maximum Sum of Sub-Arrays

ID:10626 Solved By 620 Users

The program must accept an array of **N** integers as the input. The program must print the maximum sum of sub-arrays in the given array as the output.

Boundary Condition(s):
1 <= N <= 10^6
-1000 <= Each integer value <= 1000

Input Format:
The first line contains N.
The second line contains N integers separated by a space.

Output Format:
The first line contains the maximum sum of sub-arrays in the given array.

Example Input/Output 1:
Input:
5
3 2 -2 5 -4

Output:
8

Explanation:
The sub-array with the maximum sum **8** is given below.
3 2 -2 5

Example Input/Output 2:
Input:
3
-5 -4 -6

Output:
-4

Max Execution Time Limit: 80 millisecs

```
8        int N = sc.nextInt();
```

utput - SkillRackTutor (run)

```
10
5 2 -2 -3 -4 20 -10 5 2 1
20
BUILD SUCCESSFUL (total time: 4 seconds)
```

5+2 -2+7 -3+5 -4+2 20+-2 -10+20

2 -2 -3 -4 20 -10

CurrSum = 5 7 5 2 -2 20 10 15 17 18

MaxSum = 5 7 20

```
        int N = sc.nextInt();
        int arr[] = new int[N];
        for (int index = 0; index < N; index++) {
            arr[index] = sc.nextInt();
        }
        int currSum = arr[0];
        int maxSum = arr[0];
        for (int index = 1; index < N; index++) {
            currSum = Math.max(currSum+arr[index], arr[index]);
            if(currSum > maxSum){
                maxSum = currSum;
            }
        }
        System.out.println(maxSum);

    }
```

Code:

```
import java.util.*;
public class Main
{
    public static void main(String[] args) {
        Scanner in=new Scanner(System.in);
        int n=in.nextInt();
        int arr[]=new int[n];
        for(int i=0;i<n;i++){
            arr[i]=in.nextInt();
        }
        int currSum= arr[0];
        int maxSum=arr[0];
        for(int i=1;i<n;i++){
            currSum=Math.max(currSum+arr[i],arr[i]);
            if(currSum>maxSum){
                maxSum=currSum;
            }
        }
        System.out.println(maxSum);
    }
}
```