DAY 6:

SESSION 1:

Problem 1:

Print Prime Numbers from 2 to N

ID:11088 Solved By 905 Users

The program must accept an integer **N** as the input. The program must print all the prime numbers from 2 to N (inclusive of N) as output.

Boundary Condition(s):
2 <= N <= 999999

Input Format:
The first line contains the value of N.

Output Format:
The first line contains all the prime numbers from 2 to N.

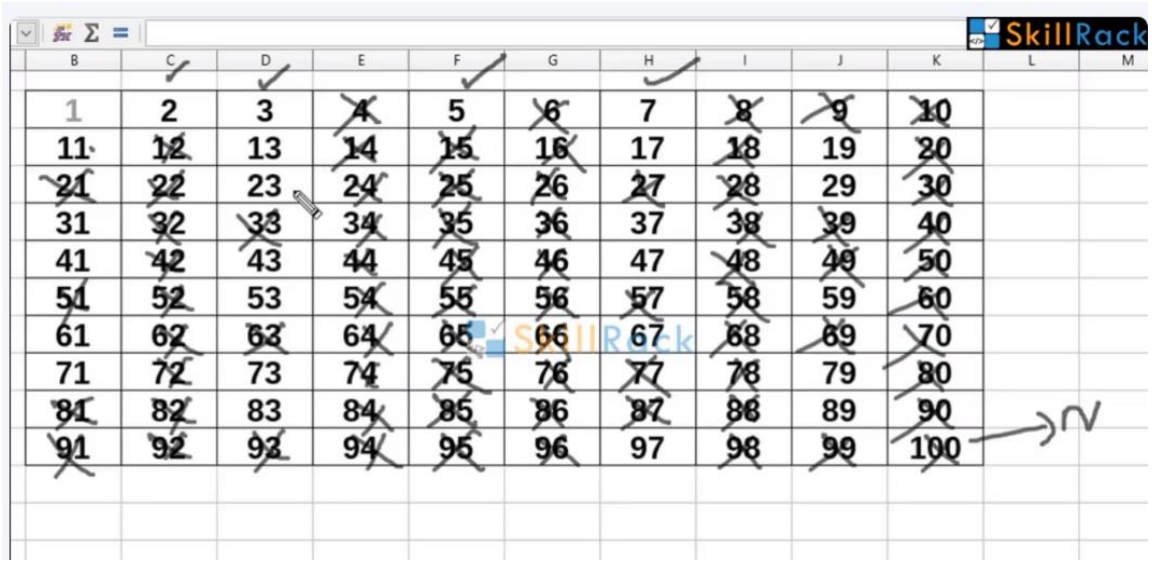
Example Input/Output 1:
Input:
11

Output:
2 3 5 7 11

Example Input/Output 2:
Input:
120

Output:
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 107 109 113

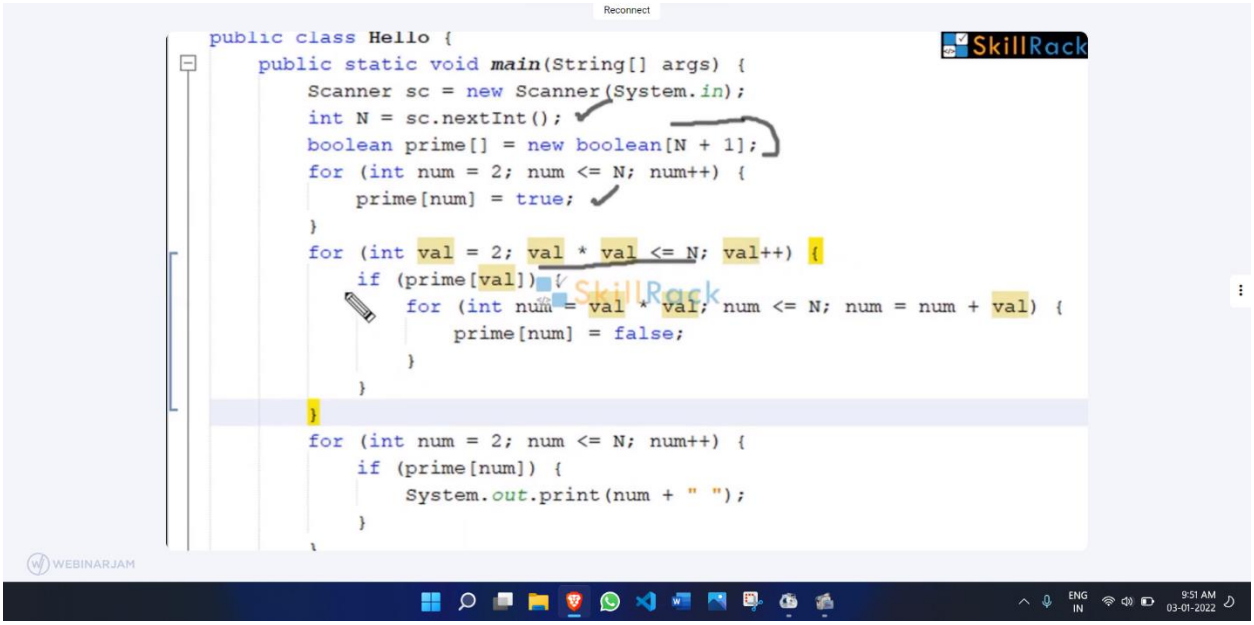
Max Execution Time Limit: 3000 millisecs



Code:

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

int main()
{
    int N;
    scanf("%d",&N);
    short primearr[N+1];
    for(int num=2;num<=N;num++){
        primearr[num]=1;
    }
    for(int val =2;val*val<=N;val++){
        if(primearr[val]==1){
            for(int num=val*val;num<=N;num=num+val){
                primearr[num]=0;
            }
        }
    }
    for(int num=2;num<=N;num++){
        if(primearr[num]==1){
            printf("%d ",num);
        }
    }
}
```



Reconnect

```
N = int(input())
prime = {}
for num in range(2,N+1):
    prime[num] = True

val = 2
while val*val <= N:
    if prime[val]:
        num = val*val;
        while num <= N:
            prime[num] = False
            num = num+val
    val = val+1

for num in range(2,N+1):
    if prime[num]:
        print(num,sep="",end=" ")
```

2 to N

Problem 2:

HCF of N Integers

ID:11089 Solved By 898 Users

The program must accept **N** integers as the input. The program must print the HCF of the N integers as the output.

Boundary Condition(s):
2 <= N <= 100
1 <= Each integer value <= 10^18

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

Output Format:
The first line contains the HCF of the N integers.

Example Input/Output 1:
Input:
4
15 20 30 50

Output:
5

Example Input/Output 2:
Input:
5
14 28 35 70 92

Output:
1

Max Execution Time Limit: 1000 millisecs

Code:

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

ULL findHCF(ULL a,ULL b){
    return b==0 ? a:findHCF(b,a%b); // Euclidean algorithm
}

int main()
{
    int N;
    scanf("%d",&N);
    ULL hcf,currNum;
    scanf("%llu",&hcf); //first input as HCF
    for(int ctr=2;ctr<=N;ctr++){
        scanf("%llu",&currNum);
        hcf=findHCF(hcf,currNum);
    }
    printf("%llu",hcf);
}
```

SESSION 2:

Problem 1:

Path Exists from Source to Destination Cell

ID:11090 Solved By 834 Users

The program must accept a matrix of size **R*C** and the indices of two cells (Source and Destination) in the matrix as the input. The matrix contains only **1's** and **0's**. The cell value **1** indicates the presence of a path. The cell value **0** indicates the presence of a stone (i.e., no path). The movement from one cell to another can be in the **left**, **right**, **bottom** and **top** directions. The program must print **yes** if there is a path from the given source cell to the destination cell. Else the program must print **no** as the output.

Boundary Condition(s):
2 <= R, C <= 50

Input Format:
The first line contains R and C separated by a space.
The next R lines, each containing C integers separated by a space.
The $(R+2)^{nd}$ line contains two integers representing the indices of the source cell.
The $(R+3)^{rd}$ line contains two integers representing the indices of the destination cell.

Output Format:
The first line contains yes or no.

Example Input/Output 1:
Input:
4 5
1 0 1 1 0
0 1 0 1 1
1 1 0 1 0
1 1 1 1 1
1 1
1 4

Output:
yes

Explanation:
One of the possible paths from the source cell to the destination cell in the matrix is highlighted below.
1 0 1 1 0
0 **1** 0 **1 1**
1 **1** 0 **1** 0
1 **1 1 1** 1

Example Input/Output 2:
Input:
3 3
1 0 1
0 1 1
1 0 1
0 2
2 0

Output:
no

Max Execution Time Limit: 500 millisecs

Stimulation:

https://pythontutor.com/visualize.html#code=%23include%3Cstdio.h%3E%0A%23include%3Cstdlib.h%3E%0Aint%20R%3D3,C%3D3,found%3D0%3B%0Aint%20sourceR%3D0,sourceC%3D2,destR%3D2,destC%3D0%3B%0A%0Avoid%20traverse%28int%20matrix%5BR%5D%5BC%5D,int%20row,int%20col%29%7B%0A%20%20%20%20if%28row%3E%3D0%20%26%26%20row%20%3CR%20%26%26%20col%3E%3D0%20%26%26%20col%3CC%29%7B%20//boundary%20condition%0A%20%20%20%20%20%20%20%20if%28row%3D%3DdestR%20%26%26%20col%3D%3DdestC%29%7B%20//if%20reached%20%0A%20%20%20%20%20%20%20%20%20%20%20%20found%3D1%3Breturn%3B%0A%20%20%20%20%20%20%20%20%7D%0A%20%20%20%20%20%20%20%20%0A%20%20%20%20%20%20%20%20if%28matrix%5Brow%5D%5Bcol%5D%20%3D%3D0%20%7C%7C%20matrix%5Brow%5D%5Bcol%5D%3D%3D2%29%7B%20//0-%3E%20no%20path%3B%202-%3Ealready%20visited%0A%20%20%20%20%20%20%20%20%20%20%20%20return%3B%0A%20%20%20%20%20%20%20%20%7D%0A%20%20%20%20%20%20%20%20matrix%5Brow%5D%5Bcol%5D%3D2%3B%20//modifying%20as%20cell%20visited%0A%20%20%20%20%20%20%20%20traverse%28matrix,row,col-1%29%3B%20//left%0A%20%20%20%20%20%20%20%20if%28!found%29%7B%0A%20%20%20%20%20%20%20%20%20%20%20%20traverse%28matrix,row,col%2B1%29%3B%20//Right%0A%20%20%20%20%20%20%20%20%7D%0A%20%20%20%20%20%20%20%20if%28!found%29%7B%0A%20%20%20%20%20%20%20%20%20%20%20%20traverse%28matrix,row%2B1,col%29%3B%20//Bottom%0A%20%20%20%20%20%20%20%20%7D%0A%20%20%20%20%20%20%20%20if%28!found%29%7B%0A%20%20%20%20%20%20%20%20%20%20%20%20traverse%28matrix,row-1,col%29%3B%20//Top%0A%20%20%20%20%20%20%20%20%7D%0A%20%20%20%20%7D%0A%7D%0A%0Aint%20main%28%29%0A%7B%0A%0A%20%20%20%20int%20matrix%5B3%5D%5B3%5D%20%3D%20%7B%0A%20%20%20%20%7B1,%200,%201%7D,%0A%20%20%20%20%7B0,%201,%201%7D,%0A%20%20%20%20%7B1,%200,%201%7D%0A%7D%3B%0A%0A%0A%20%20%20%20if%28matrix%5BsourceR%5D%5BsourceC%5D%3D%3D0%20%7C%7C%20matrix%5BdestR%5D%5BdestC%5D%3D%3D0%29%7B%0A%20%20%20%20%20%20%20%20printf%28%22no%22%29%3Breturn%3B%0A%20%20%20%20%7D%0A%20%20%20%20traverse%28matrix,sourceR,sourceC%29%3B%20//depth%20search%0A%20%20%20%20printf%28found%3D%3D1%3F%22yes%22%3A%22no%22%29%3B%0A%20%20%20%20return%200%3B%0A%7D%0A%0A&cumulative=false&curInstr=50&heapPrimitives=nevernest&mode=display&origin=opt-frontend.js&py=c_gcc9.3.0&rawInputLstJSON=%5B%5D&textReferences=false

Code:

```
#include<stdio.h>
#include<stdlib.h>
int R,C,found=0;
int sourceR,sourceC,destR,destC;

void traverse(int matrix[R][C],int row,int col){
    if(row>=0 && row <R && col>=0 && col<C){ //boundary condition
        if(row==destR && col==destC){ //if reached
            found=1;return;
        }
```

```
        if(matrix[row][col] ==0 || matrix[row][col]==2){ //0-> no path; 2-
>already visited
            return;
        }
        matrix[row][col]=2; //modifying as cell visited
        traverse(matrix,row,col-1); //left
        if(!found){
            traverse(matrix,row,col+1); //Right
        }
        if(!found){
            traverse(matrix,row+1,col); //Bottom
        }
        if(!found){
            traverse(matrix,row-1,col); //Top
        }
    }
}

int main()
{
    scanf("%d%d",&R,&C);
    int matrix[R][C];
    for(int row=0;row<R;row++){
        for(int col=0;col<C;col++){
            scanf("%d",&matrix[row][col]);
        }
    }
    scanf("%d%d%d%d",&sourceR,&sourceC,&destR,&destC);
    if(matrix[sourceR][sourceC]==0 || matrix[destR][destC]==0){
        printf("no");return;
    }
    traverse(matrix,sourceR,sourceC); //depth search
    printf(found==1?"yes":"no");
    return 0;
}
```

Problem 2:

Iterations Count All Zero

ID:11091 Solved By 832 Users

The program must accept an integer matrix of size **R*C** and an integer **K** as the input. For each occurrence of K in the matrix, the program must replace K and all the adjacent non-zero cell values with zero which are to it's **left, right, top** and **bottom**. The program must repeat the process untill all the values become zero. The program must print how many times the process has to be performed to convert all the cell values to zero.

Boundary Condition(s):
2 <= R, C <= 50

Input Format:
The first line contains R and C separated by a space.
The next R lines, each containing C integers separated by a space.
The $(R+2)^{nd}$ line contains K.

Output Format:
The first line contains an integer representing the number of times the above process has to be performed to convert all the cell values to zero.

Example Input/Output 1:
Input:
5 5
5 6 0 5 6
1 8 8 0 2
5 5 5 0 6
4 5 5 5 0
8 8 8 8 8
6

Output:
2

Explanation:
After performing the process for the first occurrence of **6**, the matrix becomes
0 0 0 5 6
0 0 0 0 2
0 0 0 0 6
0 0 0 0 0
0 0 0 0 0
After performing the process for the second occurrence of **6**, the matrix becomes
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
Now, all the cell values in the matrix become zero.
Hence the output is 2

Example Input/Output 2:
Input:
4 5
5 0 0 5 6
1 0 8 1 0
0 5 0 0 6
4 5 0 5 2
5

Output:
4

Max Execution Time Limit: 500 millisecs

Code:

```
#include<stdio.h>
#include<stdlib.h>
int R,C,interations=0;

void traverse(int matrix[R][C],int row,int col){
    if(row>=0&&row<R && col>=0 && col<C){  //checking boundary condition and row
and column should not excceed the limit
        if(matrix[row][col]==0){ //base condition
            return;
        }
        matrix[row][col] = 0;
        traverse(matrix,row,col-1);
        traverse(matrix,row,col+1);
        traverse(matrix,row+1,col);
        traverse(matrix,row-1,col);
    }
}

int main()
{
    scanf("%d%d",&R,&C);
    int matrix[R][C];
    for(int row=0;row<R;row++){
        for(int col=0;col<C;col++){
            scanf("%d",&matrix[row][col]);
        }
    }
    int K;
    scanf("%d",&K);

    for(int row=0;row<R;row++){
        for(int col=0;col<C;col++){
            if(K==matrix[row][col]){
                interations++;
                traverse(matrix,row,col);
            }
        }
    }
    printf("%d",interations);
}
```

Session 3:

Problem 1:

Valid Mix of String Values

ID:11092 Solved By 649 Users

Given 3 string values **S1**, **S2** and **S3**, find if S3 is a valid mix of the first two string values S1 and S2. The third string value S3 is said to be a mix of the first S1 string and the second S2 string, if it can be formed by interleaving the characters of the first string and the second string in a way that maintains the left to the right order of occurrence of the characters for S1 and S2 each string. The program must print **YES** if it is a valid mix. Else the program must print **NO** as the output.

Boundary Condition(s):
1 <= Length of S1, S2 <= 1000

Input Format:
The first line contains the string S1.
The second line contains the string S2.
The third line contains the string S3.

Output Format:
The first line contains either YES or NO.

Example Input/Output 1:
Input:
mno
xyz
xmnyzo

Output:
YES

Explanation:
The order of occurrence of **xyz** and **mno** is preserved in **xmnyzo.**
Hence the output YES is printed.

Example Input/Output 2:
Input:
MANO
KON
MAKNOON

Output:
YES

Example Input/Output 3:
Input:
MANO
KON
MAKOONN

Output:
NO

Max Execution Time Limit: 2000 millisecs

Code:

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

    private static boolean isValidMix(String str1,String str2,String mix, int
index1, int index2, int mixIndex){
        while(mixIndex<mix.length()){
            if(index1<str1.length() && index2<str2.length() &&
str1.charAt(index1)==mix.charAt(mixIndex)
                                && str2.charAt(index2)== mix.charAt(mixIndex) ){
//when same char at both str1 and str2 so using backtracking
```

```
                if(isValidMix(str1,str2,mix,index1+1,index2,mixIndex+1)){   //bac
ktrack
                    return true;
                }
                else{
                    return
isValidMix(str1,str2,mix,index1,index2+1,mixIndex+1);    //backtrack
                }
            }
            else if(index1<str1.length()&& str1.charAt(index1) ==
mix.charAt(mixIndex) ){
                index1++;
                mixIndex++;
            }
            else if(index2<str2.length()&& str2.charAt(index2) ==
mix.charAt(mixIndex) ){
                index2++;
                mixIndex++;
            }
            else{
                return false;
            }
        }
        return true;
    }

    public static void main(String[] args) {
        //Your Code Here
        Scanner in=new Scanner(System.in);
        String str1 = in.nextLine();
        String str2 = in.nextLine();
        String mix = in.nextLine();
        int index1=0,index2=0,mixIndex=0;
        if(str1.length()+str2.length()!=mix.length()){
            System.out.println("NO"); return;
        }
        if(isValidMix(str1,str2,mix,index1,index2,mixIndex)){
            System.out.println("YES");
        }
        else{
            System.out.println("NO");
        }
    }
}
```

Problem 2:

Valid String Mix Reverse

ID:11093 Solved By 620 Users

Given three string values S1, S2 and S3, find if S3 is a reversed valid mix of the first two string values S1 and S2. The third string value S3 is said to be a reversed valid mix of the first S1 string and the second S2 string, if it can be formed by interleaving the characters of the first string and the second string in a way that maintains the left to the right order of occurrence of the characters for S1 and S2 each string from the end of S3 to the start of S3. The program must print YES if it is a valid mix. Else the program must print NO as the output.

Boundary Condition(s):
1 <= Length of S1, S2 <= 1000

Input Format:
The first line contains the string S1.
The second line contains the string S2.
The third line contains the string S3.

Output Format:
The first line contains either YES or NO.

Example Input/Output 1:
Input:
yummy
tasty
ytymmusaty

Output:
YES

Explanation:
The order of occurrence of yummy and tasty is preserved in ytymmusaty from the end to start.
Hence the output YES is printed.

Example Input/Output 2:
Input:
HEN
NECK
KNECHEN

Output:
YES

Example Input/Output 3:
Input:
HEN
NECK
KNCeHEN

Output:
NO

Max Execution Time Limit: 4000 millisecs

Code:

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

    private static boolean isValidMix(String str1,String str2,String mix,int
index1,int index2,int mixIndex){
        while(mixIndex<mix.length()){
            if(index1<str1.length() && index2<str2.length() &&
str1.charAt(index1)==mix.charAt(mixIndex)
                                && str2.charAt(index2)==mix.charAt(mixIndex)){
//when same char at both str1 and str2 so using backtracking
                if(isValidMix(str1,str2,mix,index1+1,index2,mixIndex+1)){
//backtrack
```

```
                    return true;
                }
                else{
                    return
isValidMix(str1,str2,mix,index1,index2+1,mixIndex+1);  //backtrack
                }
            }
            else if(index1<str1.length()&&
str1.charAt(index1)==mix.charAt(mixIndex)){
                index1++;
                mixIndex++;
            }
            else if(index2<str2.length()&&
str2.charAt(index2)==mix.charAt(mixIndex)){
                index2++;
                mixIndex++;
            }
            else{
                return false;
            }
        }
        return true;
    }

    public static void main(String[] args) {
        //Your Code Here
        Scanner in = new Scanner(System.in);
        String str1 = in.nextLine();
        String str2 = in.nextLine();
        String mix = in.nextLine();
        str1=new StringBuilder(str1).reverse().toString();
        str2=new StringBuilder(str2).reverse().toString();
        int index1=0,index2=0,mixIndex=0;
        if(str1.length()+str2.length()!= mix.length()){
            System.out.println("NO"); return;
        }
        if(isValidMix(str1,str2,mix,index1,index2,mixIndex)){
            System.out.println("YES");
        }
        else{
            System.out.println("NO");
        }

    }
}
```