**LAB1**

**Inlab1(Binary Pattern)**

import java.util.\*;

class BiPattern

{

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

String s=sc.next();

int len=s.length();

char c[]=new char[len];

c=s.toCharArray();

prints(c,0);

}

public static void prints(char c[],int i)

{

if(c.length==i)

{

System.out.println(c);

return;

}

if(c[i]=='?')

{

c[i]='0';

prints(c,i+1);

c[i]='1';

prints(c,i+1);

c[i]='?';

}

else prints(c,i+1);

}

}

**INLAB2(stacks of cylinders)**

import java.util.\*;

class Cylinders

{

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

int n1=sc.nextInt();

int n2=sc.nextInt();

int n3=sc.nextInt();

Stack<Integer> stk1= new Stack<Integer>();

Stack<Integer> stk2= new Stack<Integer>();

Stack<Integer> stk3= new Stack<Integer>();

int sum1=0,sum2=0,sum3=0;

for(int i=0;i<n1;i++)

{

int j=sc.nextInt();

stk1.push(j);

sum1+=j;

//System.out.println("sum1"+sum1);

}

for(int i=0;i<n2;i++)

{

int j=sc.nextInt();

stk2.push(j);

sum2+=j;

//System.out.println("sum1"+sum2);

}

for(int i=0;i<n3;i++)

{

int j=sc.nextInt();

stk3.push(j);

sum3+=j;

//System.out.println("sum1"+sum3);

}

while(sum1!=sum2||sum2!=sum3){

if(sum1>sum2 &&sum1>sum3)

{

int p=stk1.get(0);

stk1.removeElementAt(0);

sum1-=p;

//System.out.println("sum1"+sum1);

}

if(sum2>sum1 &&sum2>sum3)

{

int p=stk2.get(0);

stk2.removeElementAt(0);

sum2-=p;

//System.out.println("sum2"+sum2);

}

if(sum3>sum2 &&sum3>sum1)

{

int p=stk3.get(0);

stk3.removeElementAt(0);

sum3-=p;

//System.out.println("sum3"+sum3);

}

}

System.out.println(sum1);

}

}

**INLAB 3(islands)**

***ISLANDS.JAVA***

import java.util.\*;  
import java.lang.\*;  
import java.io.\*;  
  
public class Islands{  
int ROW,COL;  
public Islands(int row,int col)  
{  
this.ROW=row;  
this.COL=col;  
}  
boolean isSafe(int M[][], int row, int col,boolean visited[][])  
{  
return (row >= 0) && (row < ROW) && (col >= 0) && (col < COL) && (M[row][col] == 1 && !visited[row][col]);  
}  
void DFS(int M[][], int row, int col, boolean visited[][])  
{  
int rowNbr[] = new int[] { -1, 0,0,1};  
int colNbr[] = new int[] { 0,-1,1,0};  
  
visited[row][col] = true;  
  
// Recur for all connected neighbours  
for (int k = 0; k <4; ++k)  
if (isSafe(M, row + rowNbr[k], col + colNbr[k], visited))  
DFS(M, row + rowNbr[k], col + colNbr[k], visited);  
}  
  
int countIslands(int M[][])  
{  
boolean visited[][] = new boolean[ROW][COL];  
int count = 0;  
for (int i = 0; i < ROW; ++i)  
for (int j = 0; j < COL; ++j)  
if (M[i][j] == 1 && !visited[i][j])  
{    // value 1 is not  
// visited yet, then new island found, Visit all  
// cells in this island and increment island count  
DFS(M, i, j, visited);  
++count;  
}  
  
return count;  
}  
public static void main(String[] args)  
{  
Scanner sc=new Scanner(System.in);  
int row=sc.nextInt();  
int col=sc.nextInt();  
int g[][] = new int[row][col];  
for(int i=0;i<row;i++)  
{  
for(int j=0;j<col;j++)  
{  
g[i][j]=sc.nextInt();  
}  
}  
Islands I = new Islands(row,col);  
System.out.println("Number of islands is: " + I.countIslands(g));  
}  
}

**LAB 2**

**INLAB 1(subset)**

|  |
| --- |
| arr1=set(map(int,input().split()))  arr2=set(map(int,input().split()))  if(arr2.issubset(arr1)):  print("arr2 is subeset of arr1")  else:  print("arr2 is not a subset of arr1") |
| arr1=set(map(int,input().split()))  arr2=set(map(int,input().split()))  c=0  for i in arr2:  if i in arr1:  continue  else:  c=1  break  if(c==1):  print("arr2 is not a subset of arr1")  else:  print("arr2 is a subset of arr1") |

**INLAB 2(English alphabets)**

|  |
| --- |
| class Node:  def \_\_init\_\_(self,data):  self.data=data  self.next=None  def isVowel(d):  p='aeiouAEIOU'  if d in p:  return True  else:  return False  def arrangeCV(head):  temp= head  lv= None  curr = head  if (head == None):  return None  if (isVowel(head.data)):  lv= head  else:  while (curr.next != None and not isVowel(curr.next.data)):  curr = curr.next  if (curr.next == None):  return head  lv= temp = curr.next  curr.next = curr.next.next  lv.next = head  while (curr != None and curr.next != None):  if (isVowel(curr.next.data)):  if (curr == lv):  lv = curr = curr.next  else:  p= lv.next  lv.next = curr.next  lv = lv.next  curr.next = curr.next.next  lv.next =p  else:  curr = curr.next  return temp  t=int(input())  for i in range(t):  n=int(input())  p=list(input().split())  head=Node(p[0])  tail=head  for i in range(1,n):  tail.next=Node(p[i])  tail=tail.next  temp=arrangeCV(head)  while(temp):  print(temp.data)  temp=temp.next |

**INLAB 3(Linked List Cycle)**

|  |
| --- |
| **class Node:**  **def \_\_init\_\_(self,data):**  **self.data=data**  **self.next=None**  **def detectLoop(head):**  **s=set()**  **temp=head**  **while(temp):**  **if(temp in s):**  **return True**  **s.add(temp)**  **temp=temp.next**  **return False**  **p=list(input().split())**  **head=Node(p[0])**  **tail=head**  **for i in range(1,len(p)):**  **tail.next=Node(p[i])**  **tail=tail.next**  **head.next.next.next=head**  **if(detectLoop(head)):**  **print("loop detected")**  **else:**  **print("No loop detected")** |

**LAB 3**

**INLAB1(Pointer)**

|  |
| --- |
| #include <stdio.h> void update(int \*a,int \*b) {     \*a=\*a+\*b;     if((\*a-\*b)>\*b)     \*b=(\*a-\*b)-\*b;     else     \*b=\*b-(\*a-\*b); } int main() {     int a,b;     scanf("%d %d",&a,&b);     update(&a,&b);     printf("%d %d",a,b);     return 0; } |

**INLAB 2(length of longest sub string)**

|  |
| --- |
| import java.util.\*; public class Main { public static void main(String[] args) { Scanner sc=new Scanner(System.in); String s=sc.next(); int c[]=new int[26]; int len=0,p=0; for(int i=0;i<s.length();i++) {    if(c[s.charAt(i)-'a']==0)    c[s.charAt(i)-'a']++;    else    {        len=Math.max(len,i-p);        p=i;        i--;        Arrays.fill(c,0,26,0);    } } System.out.println(len); } } |
| import java.util.\*; public class Main { public static void main(String[] args) { Scanner sc=new Scanner(System.in); int t=sc.nextInt(); String a[]=new String[t]; for(int i=0;i<t;i++) {    a[i]=sc.next(); } for(int i=0;i<t;i++) {    largest(a[i]); } }     public static  void largest(String s)     { int i=0,max=1,j; while(i<s.length()) {    int c=1;    for(j=i+1;j<s.length();j++)    {        if(((s.substring(i,j)).indexOf(s.charAt(j)))==-1)            c++;        else            break;    }    if(c>max)        max=c;          i++; } System.out.println(max); } } |

**INLAB 3(Incement by 10)**

|  |
| --- |
| #include <stdio.h> void updateVar(int \*n,int \*t) {     for(int i=0;i<\*t;i++)     n[i]+=10; } int main() {     int t,i;     scanf("%d",&t);     int a[t];     for(i=0;i<t;i++)     {         scanf("%d",&a[i]);     }     updateVar(a,&t);     for(i=0;i<t;i++)     {         printf("%d\n",a[i]);     }     return 0; } |

**LAB 4**

**INLAB 1 and INLAB 3(find min path cost)**

|  |
| --- |
| import java.util.\*; public class Main { public static void main(String[] args) {     Scanner sc=new Scanner(System.in);     int m,n;     m=sc.nextInt();     n=sc.nextInt();     int a[][]=new int[m][n];     for(int i=0;i<m;i++)     {         for(int j=0;j<n;j++)             a[i][j]=sc.nextInt();     }      for(int i=0;i<m;i++)     {         for(int j=0;j<n;j++)         {             if(i==0&&j==0)             {                 a[i][j]=a[i][j];             }             else if(i==0)             {                 a[i][j]+=a[i][j-1];             }                         else if(j==0)             {                 a[i][j]+=a[i-1][j];             }             else{                 a[i][j]+=Math.min(a[i-1][j],a[i][j-1]);             }         }     }     System.out.println(a[m-1][n-1]); } } |

**INLAB 2(cutting rod profit)**

|  |
| --- |
| def max(a, b):  return a if (a > b) else b  def cutRod(price, n):  if(n <= 0):  return 0  max\_val = -99999  for i in range(0, n):  max\_val = max(max\_val, price[i]+cutRod(price, n - i - 1))  return max\_val  p=list(map(int,input().split()))  print(cutRod(p,len(p))) |

**LAB 5**

**INLAB1(Reduce string)**

|  |
| --- |
| def reduce(i,s):      if(i<=0):          return s      if(s[i-1:i]==s[i:i+1]):          s=s[:i-1]+s[i+1:]      s=reduce(min(len(s)-1,i-1),s)      return s  s=input()  p=reduce(len(s)-1,s)  if(len(p)==0):      print("Empty String")  else:      print(p) |

**INLAB 2:(Copy to a new string)**

|  |
| --- |
| n=int(input()) l=[] res=[] for i in range(n):     l.append(input())     cost=0     s=l[i]     p=""     for j in range(len(s)):         if s[j]in p:             cost=cost+0         else:             cost+=1             p=p+s[j]     res.append(cost) for i in range(n):     print(res[i]) |

**INLAB 3(distance between two arrays)**

|  |
| --- |
| n=int(input())  a=list(map(int,input().split()))  res=n  for i in range(n):  c=0  for j in range(i+1,n):  if(a[i]==a[j]):  d=j-i  res=min(d,res)  print(res) |

**LAB 6**

**INLAB1(good bad or average)**

|  |
| --- |
| ***t=int(input()) for i in range(t):     n,x=map(int,input().split())     l=list(map(int,input().split()))     p=set(l)     if(len(p)==x):         print("Good")     elif(len(p)<x):         print("Bad")     else:         print("Average")*** |

**INLAB 2(wiggle sequence)**

|  |
| --- |
| ***t=list(map(int,input().split())) if(len(t)<2):     print(len(t)) else:     prev=0     count=len(t)     for i in range(1,len(t)):         diff=t[i]-t[i-1]         if(diff==0):             count-=1         elif((diff>0 and prev<=0) or(diff<0 and prev>=0)):             prev=diff         else:             count-=1     print(count)*** |

**INLAB 3(pivot before head)**

|  |
| --- |
| l=list(map(int,input().split())) for i in range(len(l)-1):     if(l[i]<l[i+1]):         continue     else:         break print(l[i+1]) |