In [3]:
```
#Q1. Write a program to find all pairs of an integer array whose sum is equal to a given number?

arr=list(map(int,input("enter the elements for array: ").split()))
print(arr)
key=int(input("enter a number: "))
print("the pairs are :")
for i in range(len(arr)):
    for j in range(len(arr)):
        if (arr[i]+arr[j])==key and i!=j:
            print(arr[i],",",arr[j])
```

```
enter the elements for array: 1 4 5 6 2
[1, 4, 5, 6, 2]
enter a number: 8
the pairs are :
6 , 2
2 , 6
```

In [14]:
```
#Q2. Write a program to reverse an array in place?
#In place means you cannot create a new array. You have to update the original array.

arr=list(map(int,input("enter the elements for array: ").split()))
print(arr)
for i in range(len(arr)//2):
    t=arr[i]
    arr[i]=arr[(len(arr)-1)-i]
    arr[(len(arr)-1)-i]=t
print("The updated reversed array is: ",arr)
```

```
enter the elements for array: 1 2 3 4 5 6 7 8
[1, 2, 3, 4, 5, 6, 7, 8]
The updated reversed array is:  [8, 7, 6, 5, 4, 3, 2, 1]
```

In [49]:
```
#Q3. Write a program to check if two strings are a rotation of each other?

s1=input("string1 : ")
s2=input("string2 : ")
for i in range(len(s1)):
    s3=""
    s3+=s1[-1]+s1[0:-1]
    s1=s3
    #print(s1)
    if s1==s2:
        print("Yes the two strings are in rotation")
        break
else:
    print("No the two strings are not in rotation")
```

```
string1 : hello
string2 : lohel
Yes the two strings are in rotation
```

In [116…
```
#Q4. Write a program to print the first non-repeated character from a string?

s1=input("string1 : ")
arr=list(s1)
#print(arr)

for i in range(len(arr)):
    c=1
    for j in range(len(arr)):
        if arr[i]==arr[j] and i!=j:
            c+=1
    #print(arr[i]," ",c)
    if c==1:
        print("The first non-repeated character = ",arr[i])
        break
```

```
string1 : aabbcddef
The first non-repeated character =  c
```

In [128…
```
#Q5. Read about the Tower of Hanoi algorithm. Write a program to implement it.
def toh(n , s, d, e):
    if n==1:
        print ("Move disk 1 from source",s,"to destination",d)
        return
    toh(n-1, s, e,d)
    print ("Move disk",n,"from source",s,"to destination",d)
    toh(n-1, e, d, s)

n = 3
print("Initially the drivers are tower A")
toh(n,'A','C','B')
print("Finally placed on tower C")
```

```
Initially the drivers are tower A
Move disk 1 from source A to destination C
Move disk 2 from source A to destination B
Move disk 1 from source C to destination B
Move disk 3 from source A to destination C
Move disk 1 from source B to destination A
Move disk 2 from source B to destination C
Move disk 1 from source A to destination C
Finally placed on tower C
```

In [154…
```
#Q6. Read about infix, prefix, and postfix expressions. Write a program to convert postfix to prefix expression.
def pre(post_exp):

    stack=[]
    length = len(post_exp) #reading from left to right
    for i in range(length):
        #if operator
            if (post_exp[i]=="+" or post_exp[i]=="-" or post_exp[i]=="*" or post_exp[i]=="/"):
                o1 = stack[-1]
                stack.pop()
                o2 = stack[-1]
                stack.pop()
                #print(i,"--------------",o1,o2)
                t = post_exp[i] + o2 + o1
                stack.append(t)
          # if operand
            else:
                stack.append(post_exp[i])
    s= ""
    for i in stack:
        s+= i
    return s


post_exp=input("enter postfix expression : ")
print("Prefix expression : ", pre(post_exp))
```

```
enter postfix expression : AB+CD-/
Prefix expression :  /+AB-CD
```

In [153…
```
#Q7. Write a program to convert prefix expression to infix expression.
def infix(pre_exp):

    stack=[]
    i= len(pre_exp)-1 #reading from right to left
    while i>=0:
        #if operator
        if (pre_exp[i]=="+" or pre_exp[i]=="-" or pre_exp[i]=="*" or pre_exp[i]=="/"):
            t = "(" + stack.pop() + pre_exp[i] + stack.pop() + ")"
            stack.append(t)
          # if operand
        else:
            stack.append(pre_exp[i])
        i -= 1
    s= ""
    for i in stack:
        s+= i
    return s


pre_exp=input("enter prefix expression : ")
print("infix expression : ", infix(pre_exp))
```

```
enter prefix expression : *+AB-CD
infix expression :  ((A+B)*(C-D))
```

In [186…
```
#Q8. Write a program to check if all the brackets are closed in a given code snippet.
str=input()
stack=[]
for i in range(len(str)):
    if str[i]=="[" or str[i]=="{" or str[i]=="(":
        stack.append(str[i])
        #print(stack)
    elif len(stack)!=0 and str[i]=="]" and stack[-1]=="[":
        #print(i)
        stack.pop()
    elif len(stack)!=0 and str[i]=="}" and stack[-1]=="{":
        #print(i)
        stack.pop()
    elif len(stack)!=0 and str[i]==")" and stack[-1]=="(":
        #print(i)
        stack.pop()
#print(stack)
if len(stack)==0:
    print("Brackets are balanced")
else:
    print("not balanced")
```

```
[{()}]
Brackets are balanced
```

In [191…
```
#Q9. Write a program to reverse a stack.
n=int(input("enter no. of elements "))
stack=[]
for i in range(n):
    stack.append(int(input()))
print(stack)
for i in range(len(stack)//2):
    t=stack[i]
    stack[i]=stack[(len(stack)-1)-i]
    stack[(len(stack)-1)-i]=t
print("The updated reversed stack is: ",stack)
```

```
enter no. of elements 4
1
2
3
4
[1, 2, 3, 4]
The updated reversed stack is:  [4, 3, 2, 1]
```

In [197…
```
#Q10. Write a program to find the smallest number using a stack.
n=int(input("enter no. of elements "))
stack=[]
for i in range(n):
    stack.append(int(input()))
print(stack)
min=stack[-1]
for i in range(len(stack)):
    if min>stack[i]:
        min=stack[i]
print("Smallest element is : ",min)
```

```
enter no. of elements 4
23
43
12
5
[23, 43, 12, 5]
Smallest element is :  5
```

In [ ]:

In [ ]:

In [ ]: