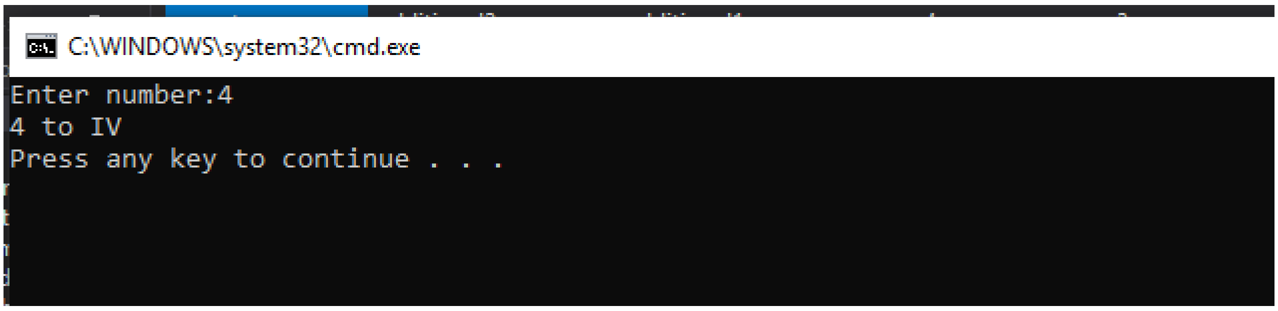
Sahil Saini Salaria
Roll No 11
Reg No 180905048

Sample

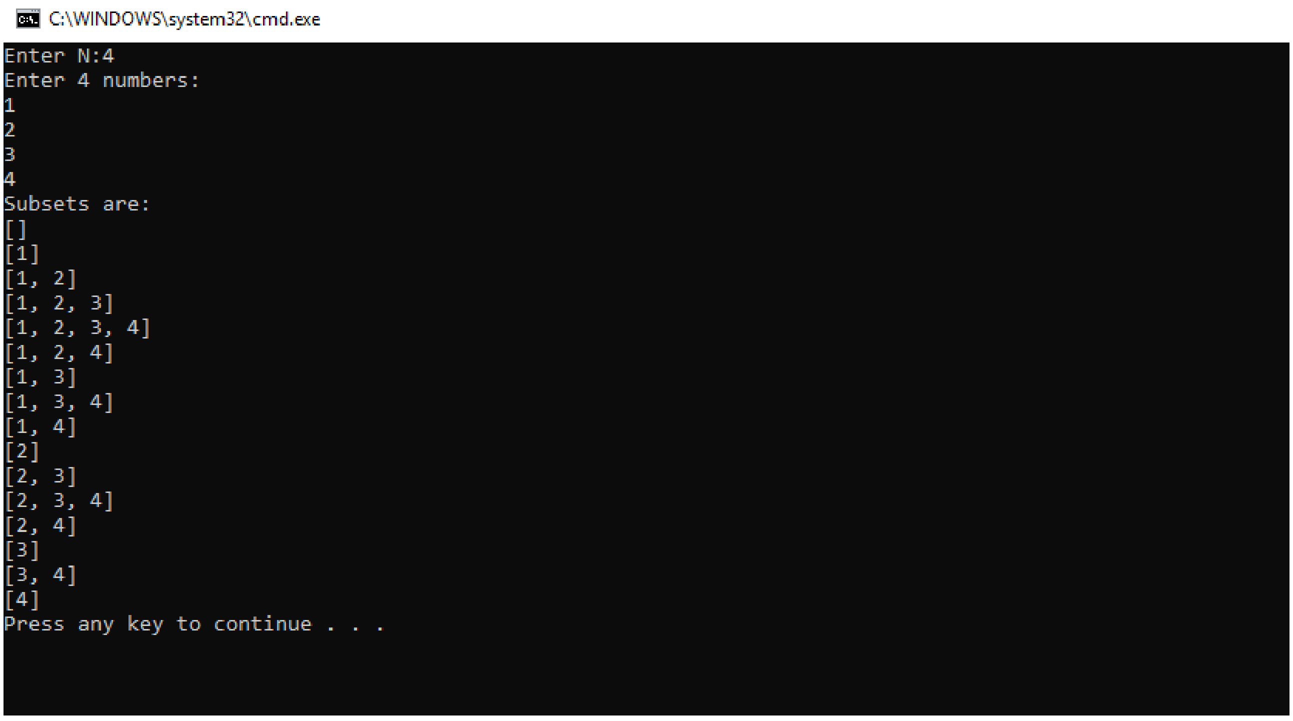
```
class myClass:
    int_arr =[1000,900,500,400,100,90,50,40,10,9,5,4,1]
    rom_arr=['M','CM','D','CD','C','XC','L','XL','X','IX','V','IV','I']
    roman_number=""
    def int_to_roman(self,number):
        i=0
        while number>0:
            for _in in range(number//self.int_arr[i]):
                self.roman_number=self.roman_number+self.rom_arr[i]
                number=number-self.int_arr[i]
            i=i+1
        return self.roman_number


myobj=myClass()
number=int(input("Enter number:"))
roman_number=myobj.int_to_roman(number)
print(str(number) +" to " + str(roman_number))
```



```
Enter number:4
4 to IV
Press any key to continue . . .
```

Q1

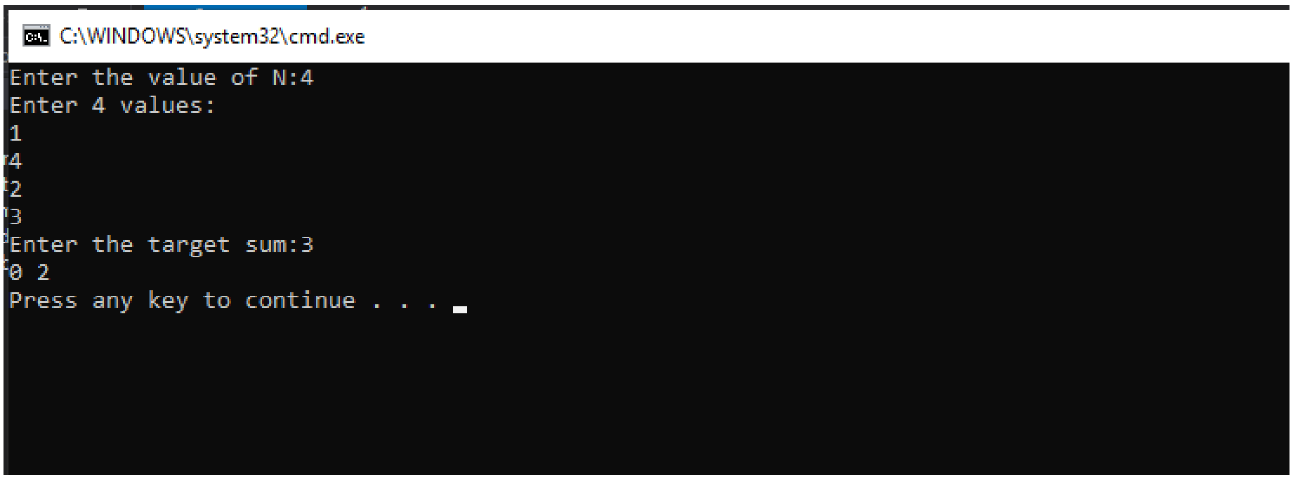
```
# import sys
# sys.setrecursionlimit(1500)

class setGenerator:
    arr=[]
    all_subsets=[]
    def get_array_input(self,N):
        print("Enter "+str(N)+" numbers:")
        for i in range(N):
            ele=int(input())
            self.arr.append(ele)

    def helper(self,i,subsets):
        # self.all_subsets.append(subsets)
        print(subsets)
        for ele in range(i,len(self.arr)):
            subsets.append(self.arr[ele])
            self.helper(ele+1,subsets)
            subsets.pop()


    def get_unique_sets(self):
        subsets=[]
        self.helper(0,subsets)
        # print(self.all_subsets)

my_obj=setGenerator()
N=int(input("Enter N:"))
my_obj.get_array_input(N)

print("Subsets are:")
my_obj.get_unique_sets()
```



```
C:\WINDOWS\system32\cmd.exe
Enter N:4
Enter 4 numbers:
1
2
3
4
Subsets are:
[]
[1]
[1, 2]
[1, 2, 3]
[1, 2, 3, 4]
[1, 2, 4]
[1, 3]
[1, 3, 4]
[1, 4]
[2]
[2, 3]
[2, 3, 4]
[2, 4]
[3]
[3, 4]
[4]
Press any key to continue . . .
```

Q2

```
class myClass:
    arr=[]
    def get_input(self,N):
        print("Enter "+str(N)+" values:")
        for i in range(N):
            ele=int(input())
            self.arr.append(ele)
    def get_two_numbers(self,target_sum):
        for i in range(len(self.arr)):
            for j in range(i+1,len(self.arr)):
                if self.arr[i]+self.arr[j]==target_sum:
                    return (i,j)
        return None


myobj=myClass()
N=int(input("Enter the value of N:"))
myobj.get_input(N)

target_sum=int(input("Enter the target sum:"))

indices=myobj.get_two_numbers(target_sum)

if indices==None:
    print("No pair found!")
else:
    print(indices[0],indices[1])
```



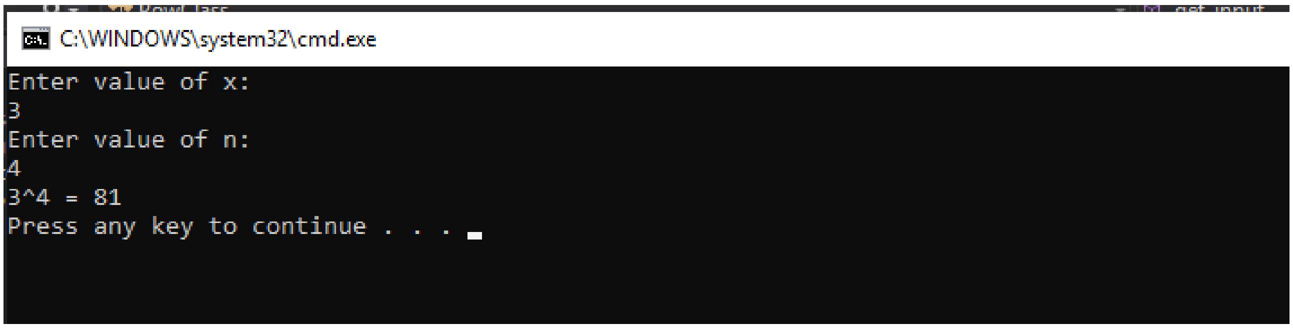
C:\WINDOWS\system32\cmd.exe

```
Enter the value of N:4
Enter 4 values:
1
4
2
3
Enter the target sum:3
0 2
Press any key to continue . . .
```

Q3

```
class PowClass:
    x=0
    n=0
    pow_var=1
    def get_input(self):
        print("Enter value of x:")
        self.x=int(input())
        print("Enter value of n:")
        self.n=int(input())

    def find_power(self):
        for i in range(self.n):
            self.pow_var=self.pow_var*self.x
        return self.pow_var
```

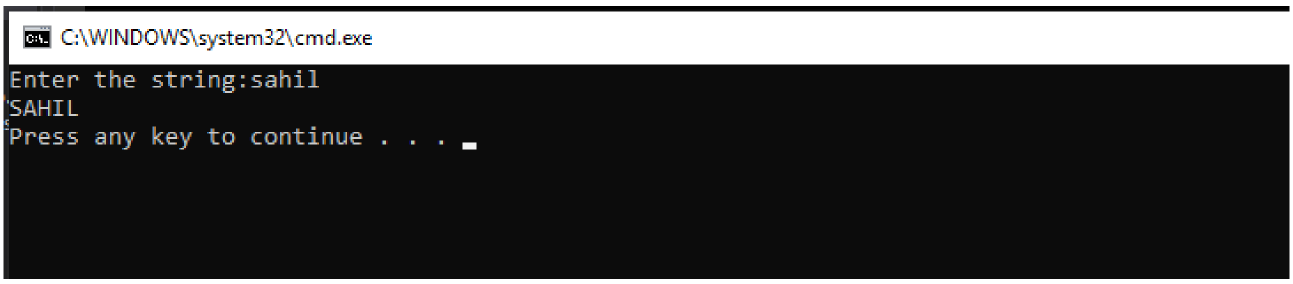
```
my_obj=PowClass()
my_obj.get_input()
pow_var=my_obj.find_power()
print(str(my_obj.x)+"^"+str(my_obj.n)+" = "+str(pow_var))
```



```
C:\WINDOWS\system32\cmd.exe
Enter value of x:
3
Enter value of n:
4
3^4 = 81
Press any key to continue . . .
```

Q4

```
class myClass:
   my_string=""

   def get_String(self):
      self.my_string=input("Enter the string:")

   def print_String(self):
      new_str=self.my_string.upper()
      print(new_str)

my_obj=myClass()

my_obj.get_String()
my_obj.print_String()
```

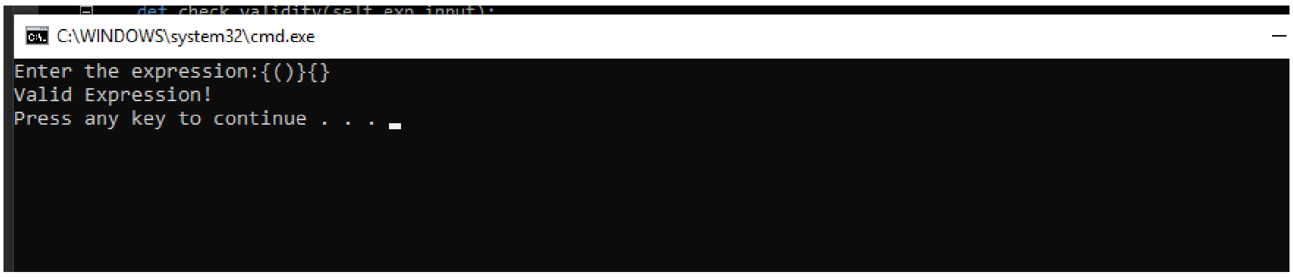


```
C:\WINDOWS\system32\cmd.exe
Enter the string:sahil
SAHIL
Press any key to continue . . .
```

ADDITIONALS

Additional 1

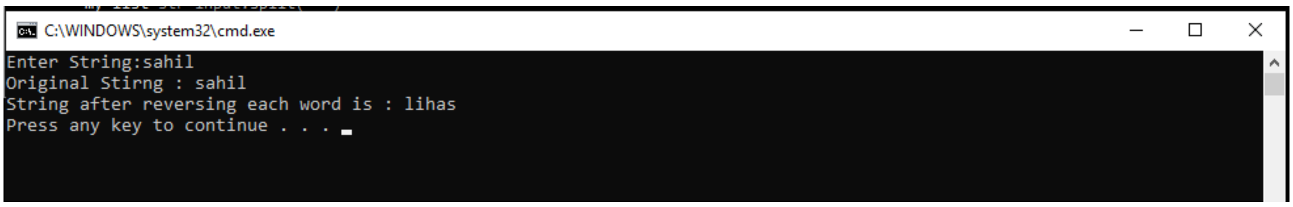
```
class Validity:

   def check_validity(self,exp_input):
      stack=[]
      exp_arr=['(','{','[']
        for ele in exp_input:
            if ele in exp_arr:
                stack.append(ele)
            elif (ele==')' and stack[len(stack)-1]!='(') or (ele=='}' and stack[len(stack)-1]!='{') or
(ele==']' and stack[len(stack)-1]!='['):
                return False
            else:
                stack.pop()

        if len(stack) !=0:
            return False

        return True

exp_input=input("Enter the expression:")

my_obj=Validity()

if my_obj.check_validity(exp_input):
    print("Valid Expression!")
else:
    print("Invalid Expression!")
```



```
Enter the expression:{()}{}
Valid Expression!
Press any key to continue . . .
```

Additional 2

```
class Reverse:

    def reverse_string_by_words(self,str_input):
        my_list=str_input.split(' ')
        new_str=""
        for ele in my_list:
            new_str=new_str+ ele[::-1]+" "
        return new_str[:len(new_str)-1]



str_input=input("Enter String:")
print("Original Stirng : "+str_input)

my_obj=Reverse()
str_input=my_obj.reverse_string_by_words(str_input)

print("String after reversing each word is : "+str_input)
```



```
Enter String:sahil
Original Stirng : sahil
String after reversing each word is : lihas
Press any key to continue . . .
```