LOTUS VALLEY INTERNATIONAL SCHOOL: Computer Science Practical File

Pranav Verma (Roll No. 7, XII Aryabhatta)

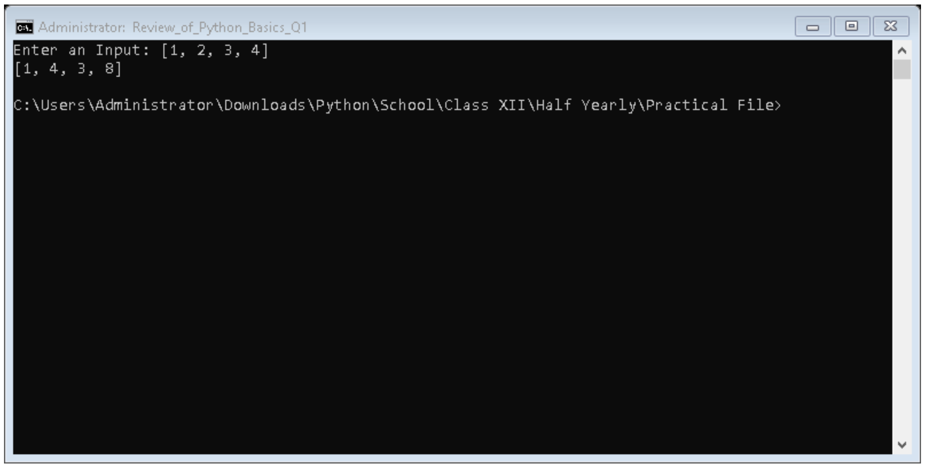
October 15, 2025

Contents

1 Review of Python Basics

Q1) Write a program to multiply an element by two, if it is an odd index for a given list containing both numbers and strings.

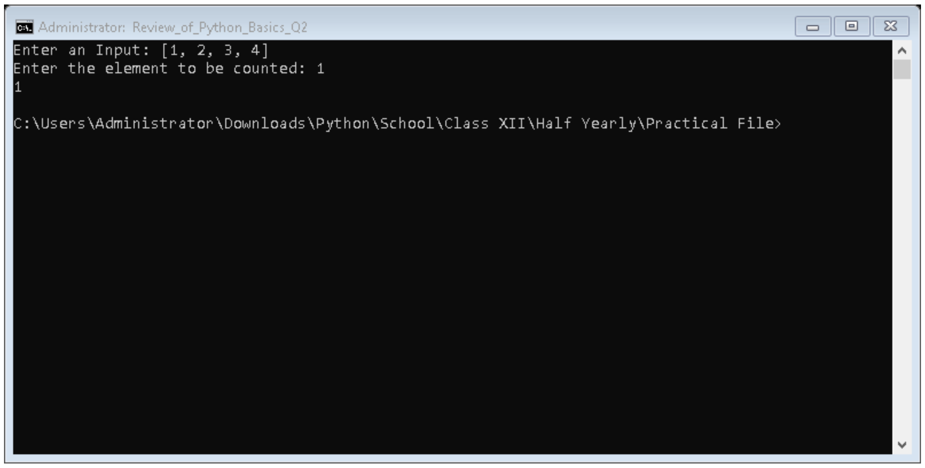
```python
l=eval(input("Enter an Input: "))
n=len(l)
for i in range(n):
    if i%2!=0:
        l[i]=l[i]*2
print(l)
```



```
Enter an Input: [1, 2, 3, 4]
[1, 4, 3, 8]

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q2) Write a program to count the frequency of an element in a list.

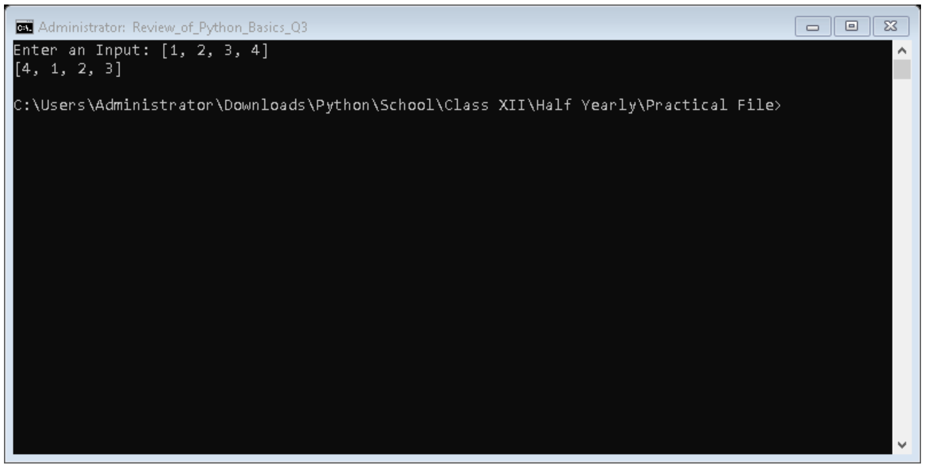
```python
l=eval(input("Enter an Input: "))
e=eval(input('Enter the element to be counted: '))
print(l.count(e))
```



```
Enter an Input: [1, 2, 3, 4]
Enter the element to be counted: 1
1

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q3) Write a program to shift elements of a list so that the first element moves to the second index and second index move to the third index and so on, and the last element shifts to the first position.

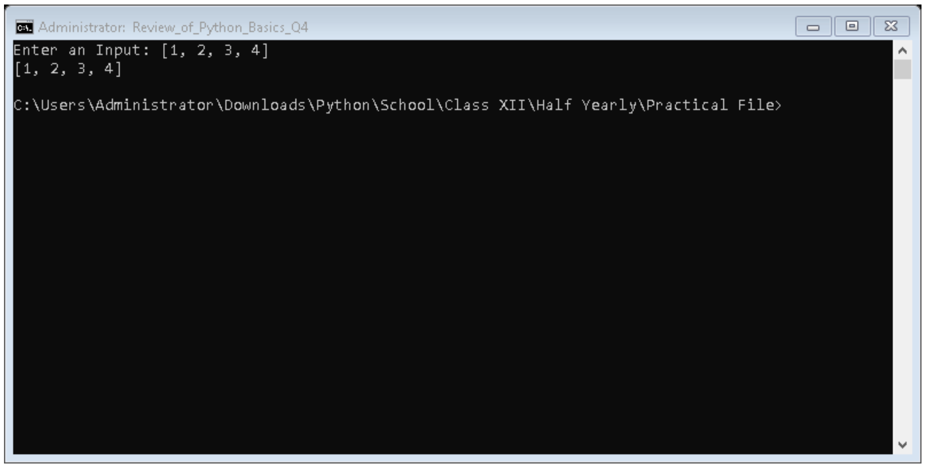
```python
l=eval(input('Enter an Input: '))
x=l[-1]
l.pop(-1)
l.insert(0,x)
print(l)
```



```
Enter an Input: [1, 2, 3, 4]
[4, 1, 2, 3]

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q4) A list NUM contains the elements: 3,25,13,6,35,8,14,45. Write a program to swap the content with the next value divisible by 5 so that the resultant list will look like: 25,3,13,35,6,8,45,14.

```python
l=eval(input('Enter an Input: '))
n=len(l)
for i in range(n):
    if l[i]%5==0:
        l[i-1],l[i]=l[i],l[i-1]
print(l)
```

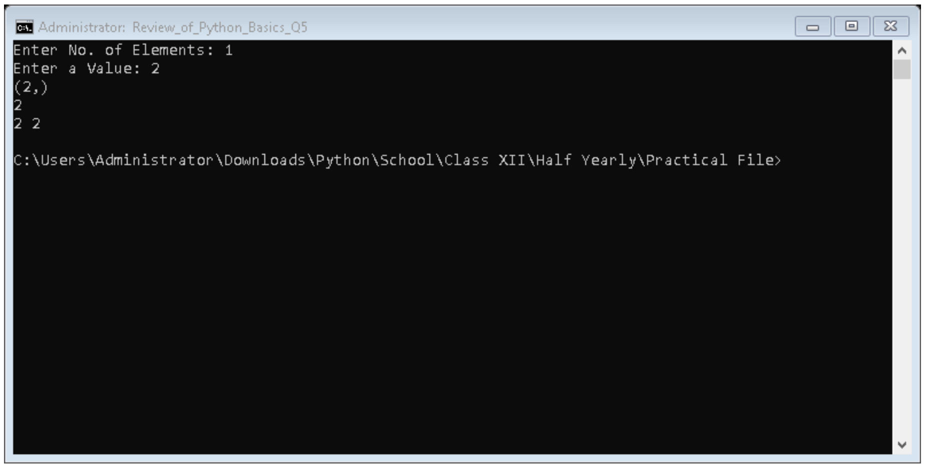


Administrator: Review_of_Python_Basics_Q4

```
Enter an Input: [1, 2, 3, 4]
[1, 2, 3, 4]

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q5) Write a program to accept values from a user in a tuple. Add a tuple to it and display its elements one by one. Also display its maximum and minimum values.

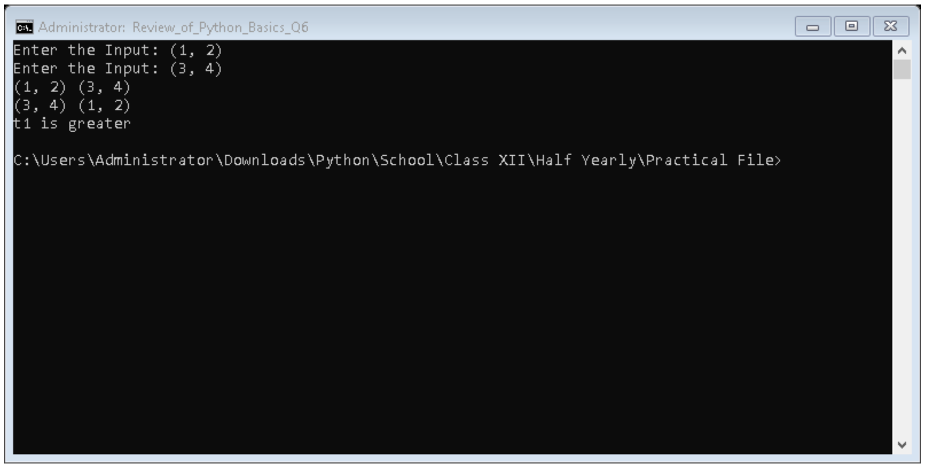
```python
n=int(input('Enter No. of Elements: '))
t=()
for i in range(n):
    x=eval(input('Enter a Value: '))
    t+=x,
print(t)
for i in t:
    print(i)
print(max(t),min(t),end='\n')
```



```
Enter No. of Elements: 1
Enter a Value: 2
(2,)
2
2 2

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q6) Write a program to input any values for two tuples. Print it, interchange it and then compare them.

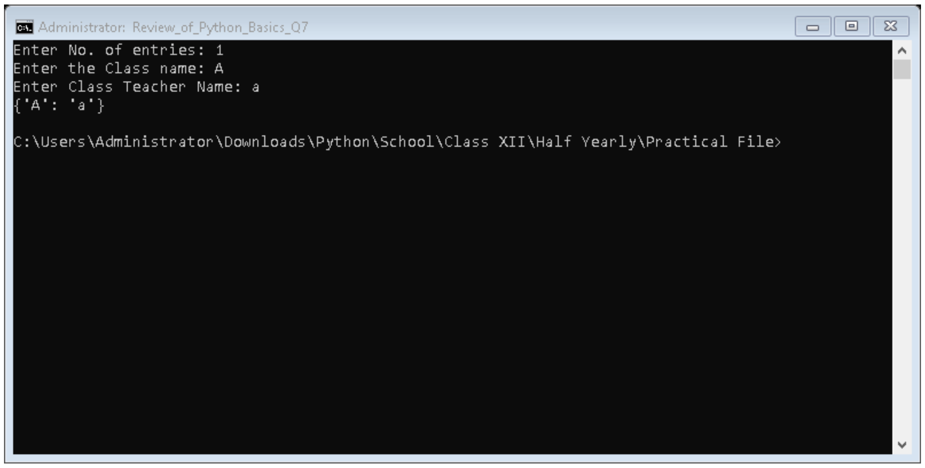
```python
t1=eval(input("Enter the Input: "))
t2=eval(input("Enter the Input: "))
print(t1, t2)
x=t2
t2=t1
t1=x
print(t1, t2)
if t1<t2:
    print('t2 is greater')
else:
    print('t1 is greater')
```



```
Enter the Input: (1, 2)
Enter the Input: (3, 4)
(1, 2) (3, 4)
(3, 4) (1, 2)
t1 is greater

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q7) Write a python program to input 'n' classes and names of class teachers to store them in a dictionary and display the same. Also accept a particular class from the user and display the name of the class teacher of that class.

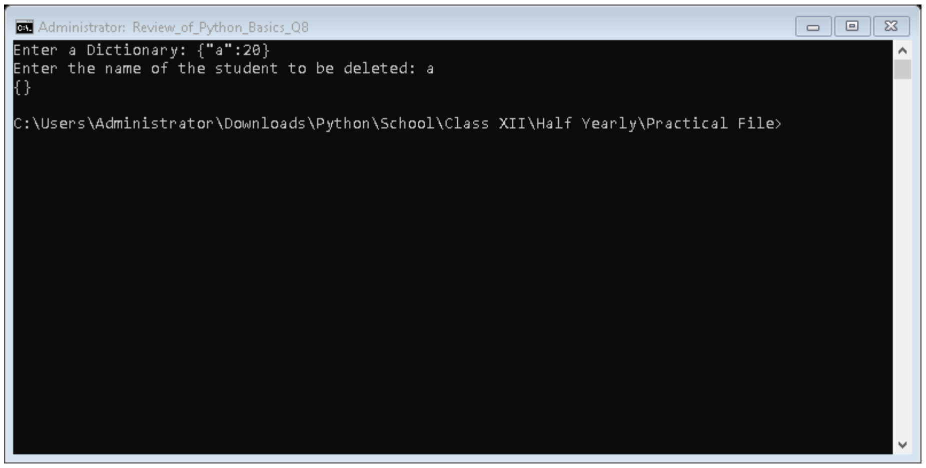
```python
n=int(input('Enter No. of entries: '))
d={}
for i in range(n):
    x=input('Enter the Class name: ')
    y=input('Enter Class Teacher Name: ')
    d[x]=y
print(d)
```



```
Administrator: Review_of_Python_Basics_Q7
Enter No. of entries: 1
Enter the Class name: A
Enter Class Teacher Name: a
{'A': 'a'}

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q8) Write a program to store students names and their percentage in a dictionary and delete a particular student name from the dictionary. Also display the dictionary after deletion.

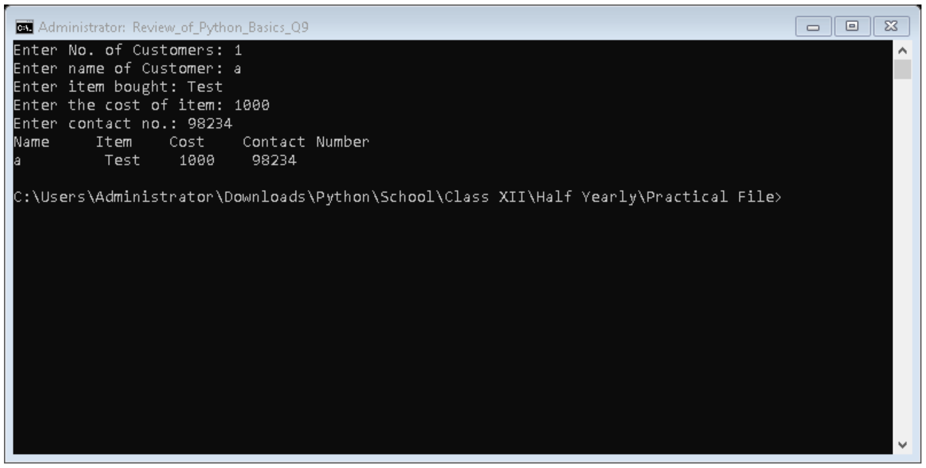
```python
d=eval(input('Enter a Dictionary: '))
x=input('Enter the name of the student to be deleted: ')
if x in d:
    del d[x]
print(d)
```



```
Enter a Dictionary: {"a":20}
Enter the name of the student to be deleted: a
{}

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q9) Write a Python program to input names of 'n' customers and their details like items bought, cost and phone number, etc., store them in a dictionary and display all the details in a tabular form.

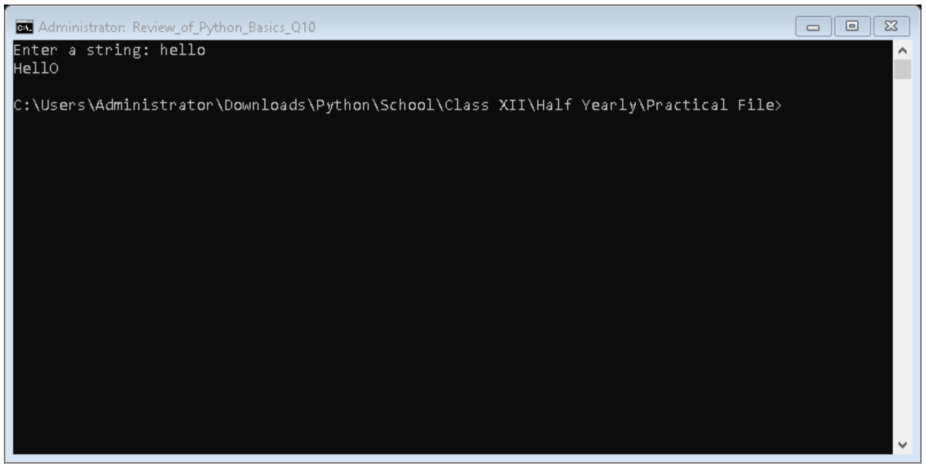
```python
d={}
n=int(input('Enter No. of Customers: '))
for i in range(n):
    name=input('Enter name of Customer: ')
    item=input('Enter item bought: ')
    cost=eval(input('Enter the cost of item: '))
    ph_no=int(input('Enter contact no.: '))
    d[name]=[item,cost,ph_no]
print('Name','\t Item','\t Cost','\t Contact Number')
for i in d:
    print(i,'\t ', d[i][0],'\t ',d[i][1],'\t ',d[i][2])
```



```
Enter No. of Customers: 1
Enter name of Customer: a
Enter item bought: Test
Enter the cost of item: 1000
Enter contact no.: 98234
Name     Item    Cost    Contact Number
a         Test    1000    98234

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q10) Write a Python program to capitalize the first and last letters of each word of a given string.

```python
s=input('Enter a string: ')
j=''
x=''
l=s.split()
for i in l:
    j=i[0].upper()+i[1:-1]+i[-1].upper()
    x+=j+''
print(x)
```

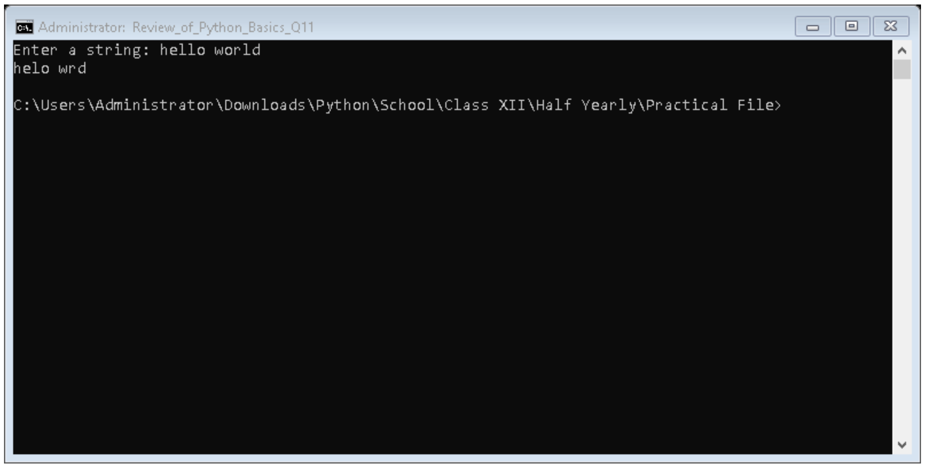


Administrator: Review_of_Python_Basics_Q10

```
Enter a string: hello
HellO

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q11) Write a Python program to remove duplicate characters of a given string.

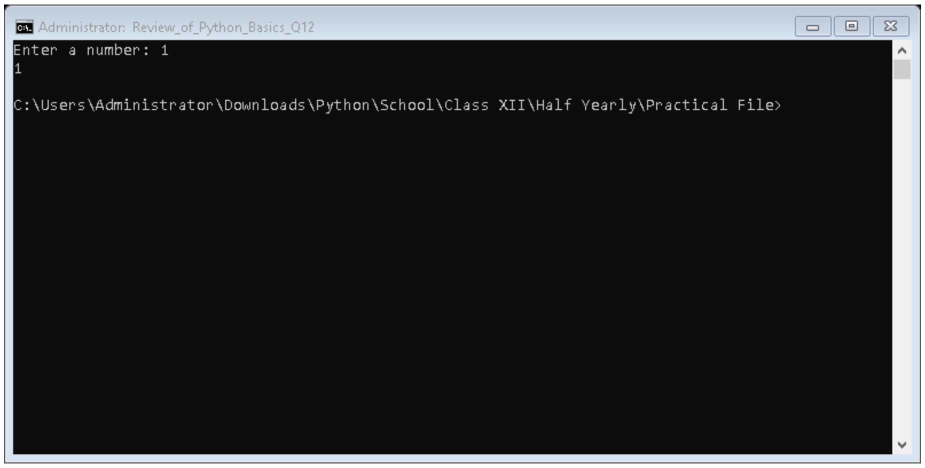
```python
s=input('Enter a string: ')
x=''
for i in s:
    if i not in x:
        x+=i
print(x)
```



```
Administrator: Review_of_Python_Basics_Q11
Enter a string: hello world
helo wrd

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q12) Write a Python program compute the sum of digits of a given number

```python
x=int(input('Enter a number: '))
s=0
while x>0:
    b=x%10
    s+=b
    x=x//10
print(s)
```

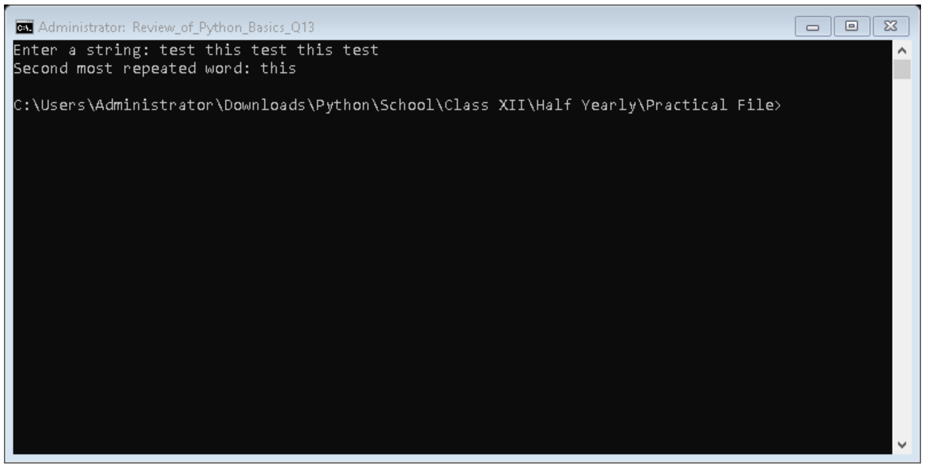


```
Administrator: Review_of_Python_Basics_Q12
Enter a number: 1
1

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q13) Write a Python program to find the second most repeated word in a given string.

```python
s = input("Enter a string: ")
words = s.split()
```

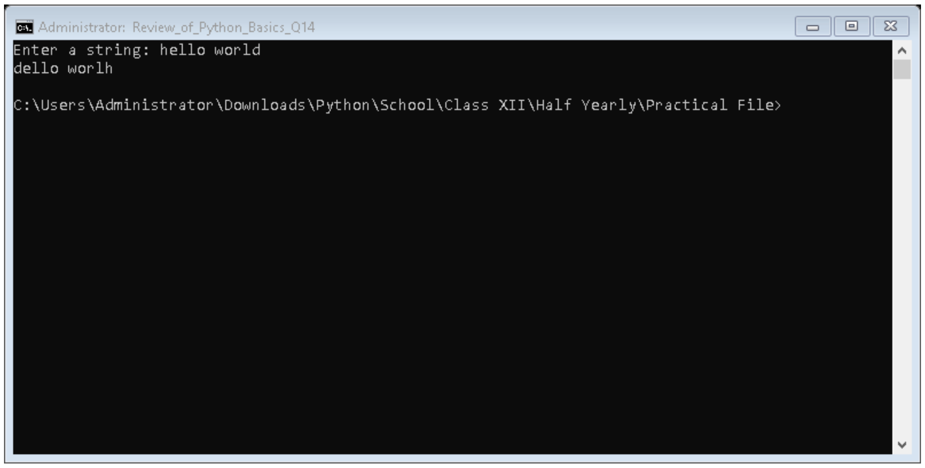
```python
freq = {}
for word in words:
    freq[word] = freq.get(word, 0) + 1
sorted_freq = sorted(freq.items(), key=lambda x: x[1], reverse=True
    )
if len(sorted_freq) < 2:
    print("No second most repeated word found.")
else:
    print("Second most repeated word:", sorted_freq[1][0])
```



```
Enter a string: test this test this test
Second most repeated word: this

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q14) Write a Python program to change a given string to a new string where the first and last string have been exchanged.

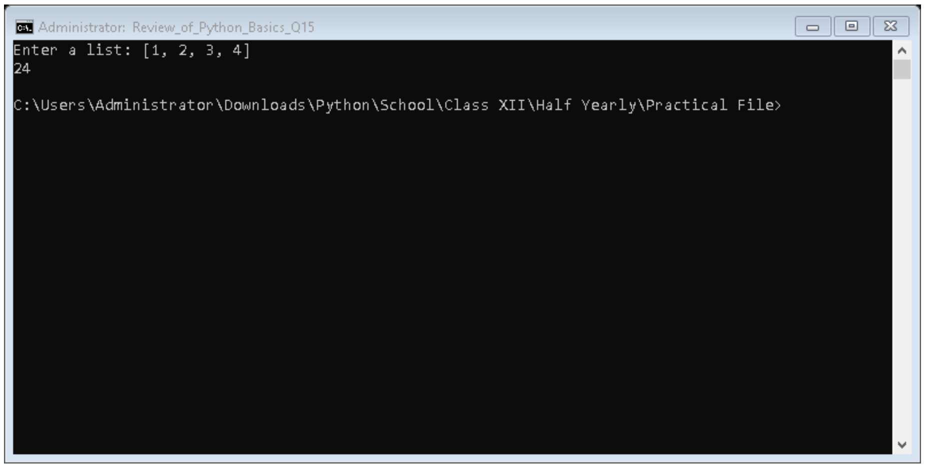
```python
s=input('Enter a string: ')
x=len(s)
a=s[-1]+s[1:x-1]+s[0]
print(a)
```



```
Administrator: Review_of_Python_Basics_Q14
Enter a string: hello world
dello worlh

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q15) Write a Python program to multiply all the elements in a list.

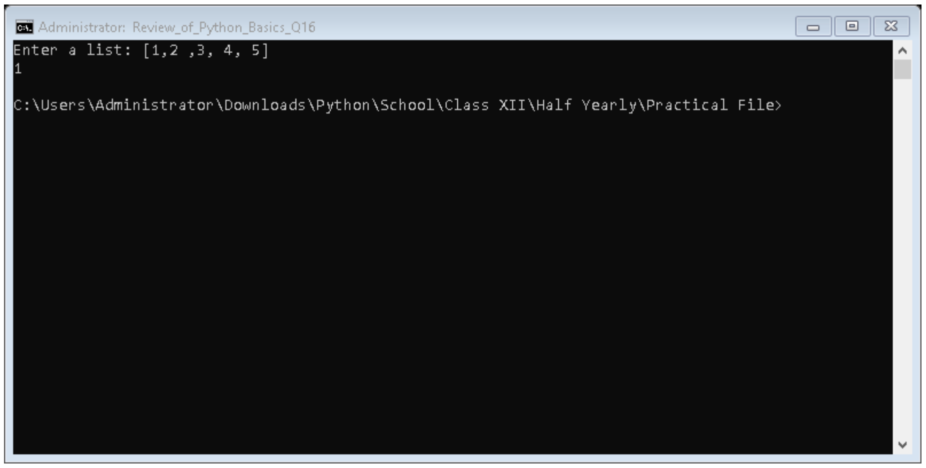
```python
l=eval(input('Enter a list: '))
p=1
for i in l:
    p*=i
print(p)
```



```
Administrator: Review_of_Python_Basics_Q15
Enter a list: [1, 2, 3, 4]
24

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q16) Write a Python program to get the smallest number from a list.

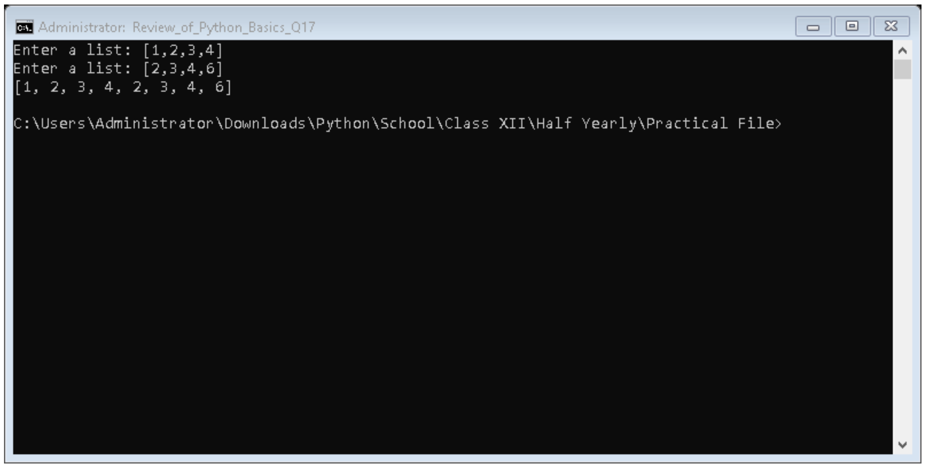
```python
l=eval(input('Enter a list: '))
print(min(l))
```



```
Enter a list: [1,2 ,3, 4, 5]
1

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q17) Write a Python program to append a list to the second list.

```python
l1=eval(input('Enter a list: '))
l2=eval(input('Enter a list: '))
l1.extend(l2)
print(l1)
```

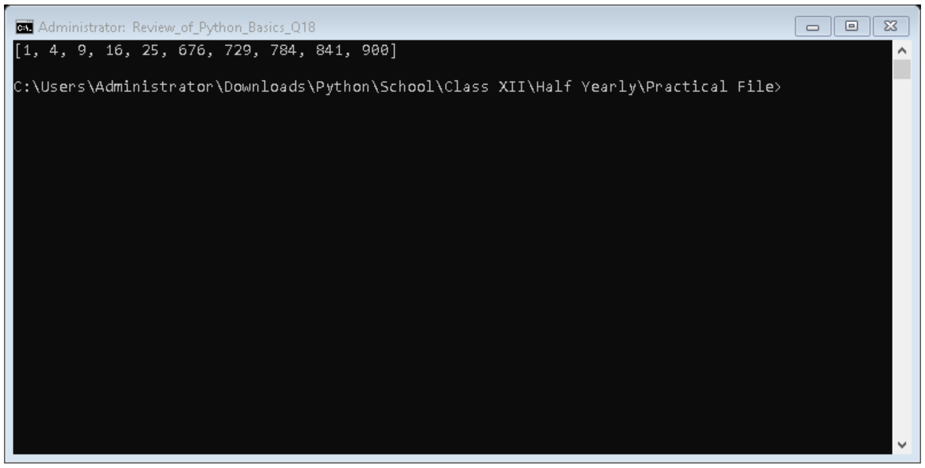


```
Enter a list: [1,2,3,4]
Enter a list: [2,3,4,6]
[1, 2, 3, 4, 2, 3, 4, 6]

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q18) Write a Python program to generate and print a list of first five and last five elements where the values are square of numbers between one and 30 (both included).

```python
l=[]
for i in range(1,31):
    l.append(i**2)
```

```
x=l[:5]+l[-5:]
print(x)
```

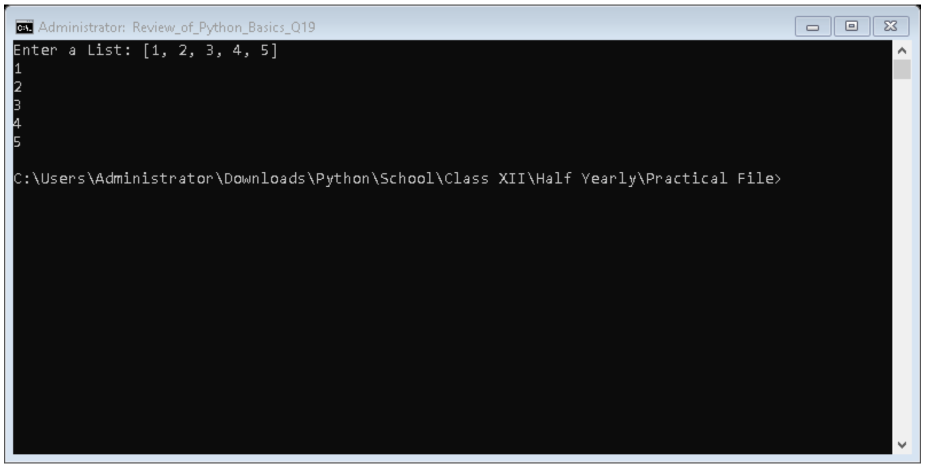


Administrator: Review_of_Python_Basics_Q18

```
[1, 4, 9, 16, 25, 676, 729, 784, 841, 900]

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q19) Write a Python program to get unique values from a list.

```
l=eval(input('Enter a List: '))
for i in l:
    if l.count(i)==1:
        print(i)
```

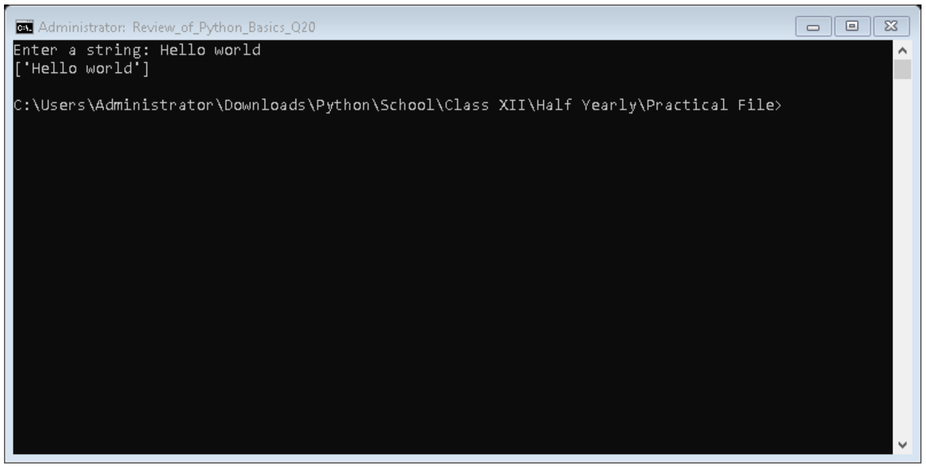


Administrator: Review_of_Python_Basics_Q19

```
Enter a List: [1, 2, 3, 4, 5]
1
2
3
4
5

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q20) Write a python program to convert a string to a list.

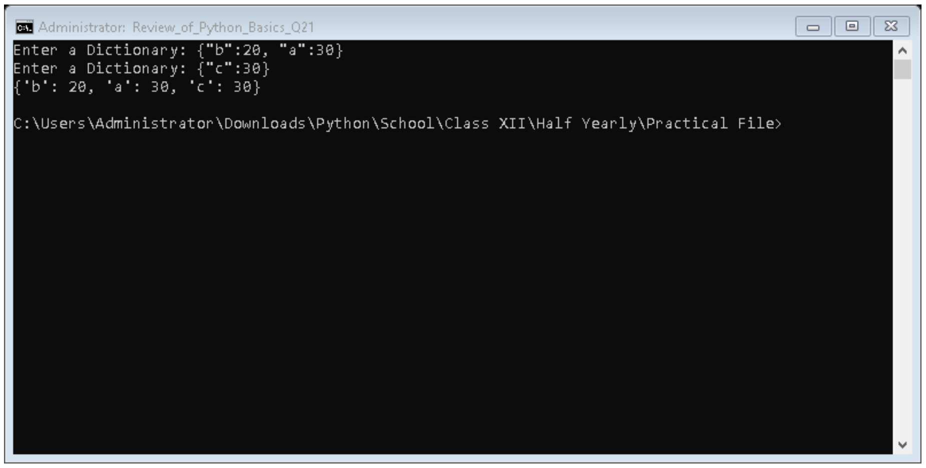
```
s=input('Enter a string: ')
l=[]
l.append(s)
print(l)
```



```
Enter a string: Hello world
['Hello world']

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q21) Write a Python script to concatenate the following dictionaries to create a new one: d1: {'A':1, 'B':2, 'C': 3}, d2: {'D':4}, Output should be: {'A':1, 'B':2, 'C':3, 'D':4}

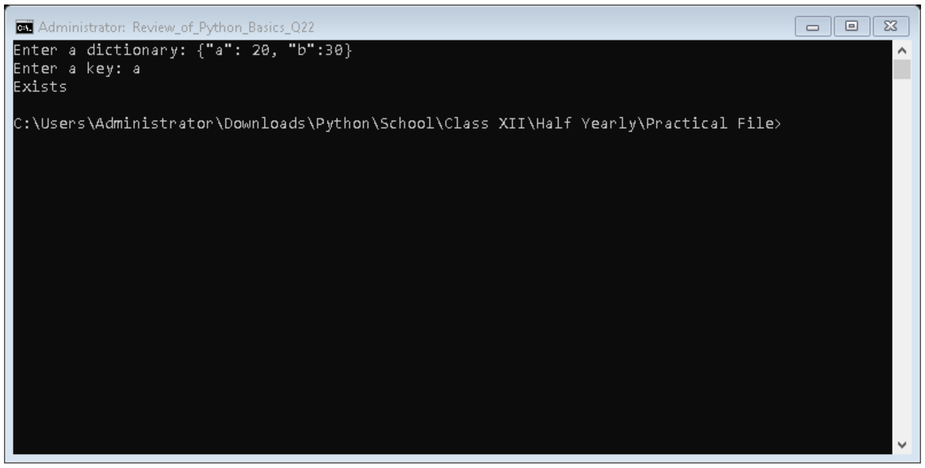
```python
d1=eval(input("Enter a Dictionary: "))
d2=eval(input("Enter a Dictionary: "))
d1.update(d2)
print(d1)
```



```
Enter a Dictionary: {"b":20, "a":30}
Enter a Dictionary: {"c":30}
{'b': 20, 'a': 30, 'c': 30}

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q22) Write a Python script to check if a given key already exists in a dictionary.

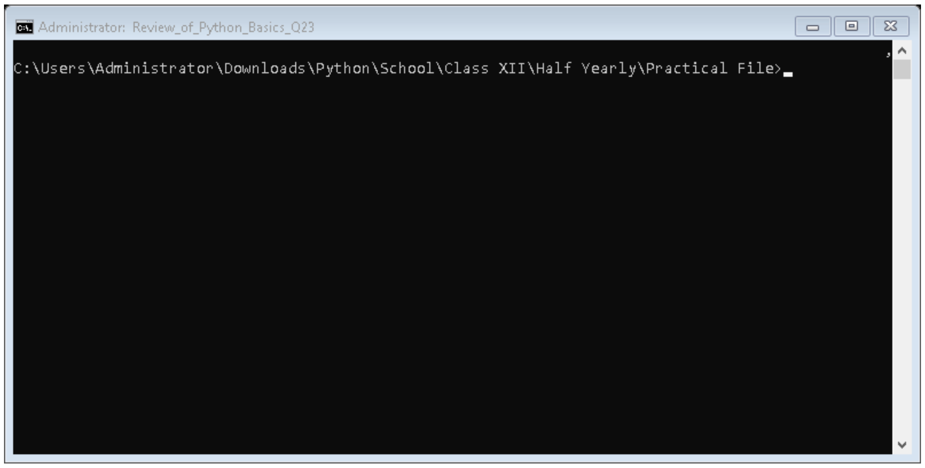
```python
d=eval(input('Enter a dictionary: '))
x=input('Enter a key: ')
if x in d:
    print('Exists')
else:
    print("Key Does Not Exist")
```



```
Enter a dictionary: {"a": 20, "b":30}
Enter a key: a
Exists

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q23) Write a Python script to print a dictionary where the keys are numbers between 1 and 15 (both included) and the values are square of keys.

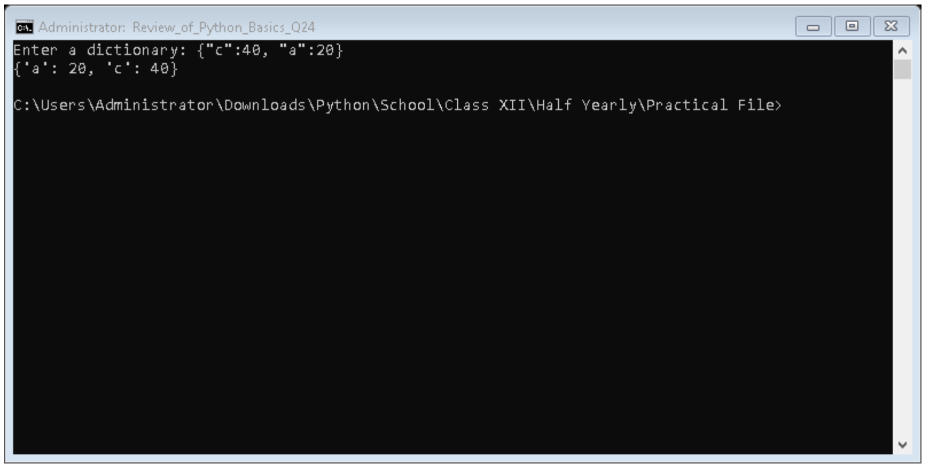
```
d={}
for i in range(1,16):
    d[i]=i**2
print(d)
```



```
C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>_
```

Q24) Write a Python program to sort a dictionary by key.

```python
d=eval(input('Enter a dictionary: '))
print(dict(sorted(d.items())))
```

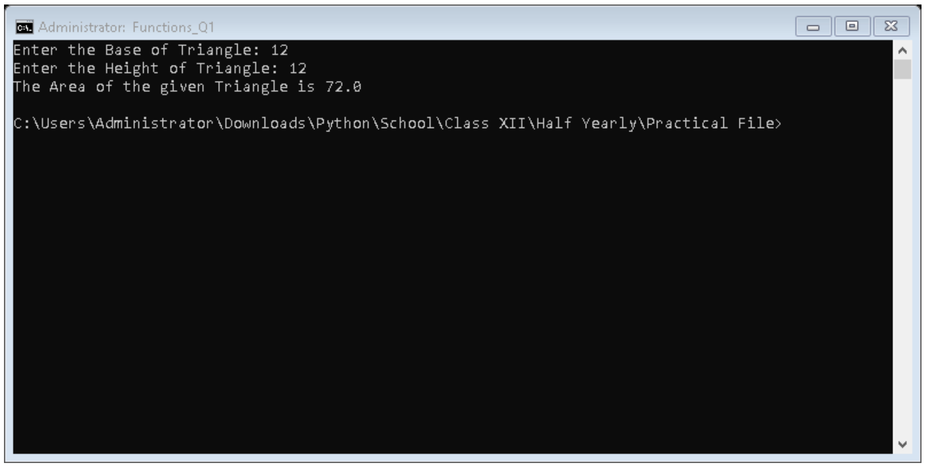

Administrator: Review_of_Python_Basics_Q24

```
Enter a dictionary: {"c":40, "a":20}
{'a': 20, 'c': 40}

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

2 Functions

Q1) Write a function, calculate_area(), that takes base and height as input arguments and returns the area of a triangle as an output. The formula used is: Triangle Area = 1/2 * base * height

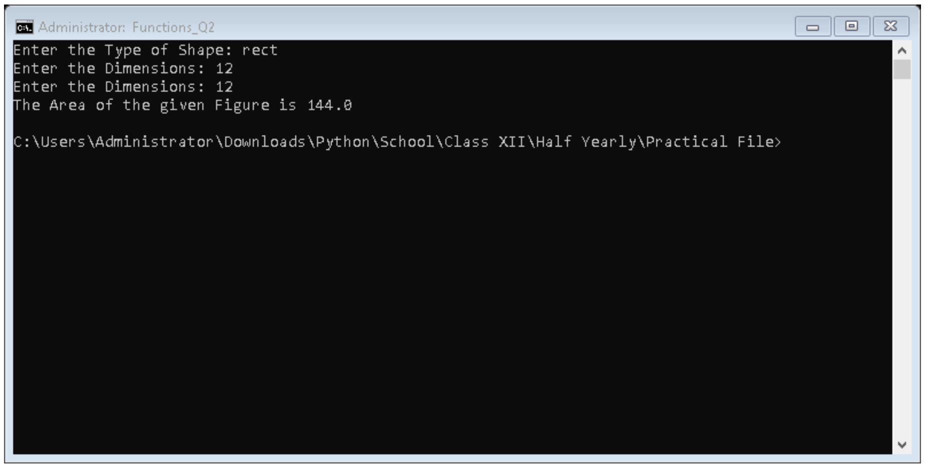
```python
def calculate_area(B,H):
    A=(1/2)*B*H
    return A
b=float(input("Enter the Base of Triangle: "))
h=float(input("Enter the Height of Triangle: "))
print("The Area of the given Triangle is",calculate_area(b,h))
```


Administrator: Functions_Q1

```
Enter the Base of Triangle: 12
Enter the Height of Triangle: 12
The Area of the given Triangle is 72.0

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q2) Modify the function given in the previous question to take a third parameter called shape type. Shaped type should be either triangle or rectangle. Based on the shape, it should calculate the area.

```python
def calculate_area(Shape_Type,B,H):
    if Shape_Type=="Rectangle":
        A=B*H
    elif Shape_Type=="Triangle":
        A=(1/2)*B*H
    return A
shape=input("Enter the Type of Shape: ")
if shape.lower() in ("rect","rectangle","r","sqaure","s"):
    shape="Rectangle"
elif shape.lower() in ("triangle","t","tri"):
    shape="Triangle"
else:
    print("Shape NOT DEFINED")
    quit()
b=float(input("Enter the Dimensions: "))
h=float(input("Enter the Dimensions: "))
print("The Area of the given Figure is",calculate_area(shape,b,h))
```

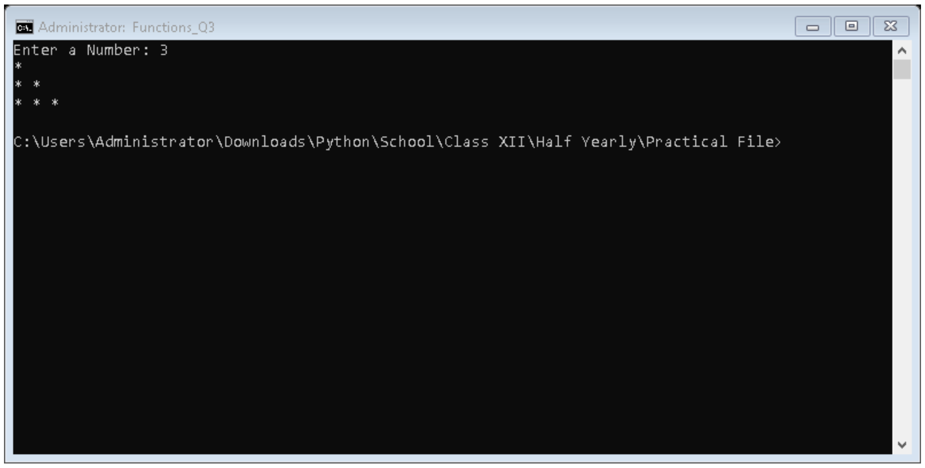


```
Enter the Type of Shape: rect
Enter the Dimensions: 12
Enter the Dimensions: 12
The Area of the given Figure is 144.0

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q3) Write a function, print_pattern(), that takes integer number as argument and print the following pattern if the input is 3: *, **, *** If the input is 4, then it should print: *, **, ***, ****.

```python
def pattern(N):
    for i in range(1,N+1):
        for j in range(i):
            print("*",end=" ")
        print()
n=int(input("Enter a Number: "))
```

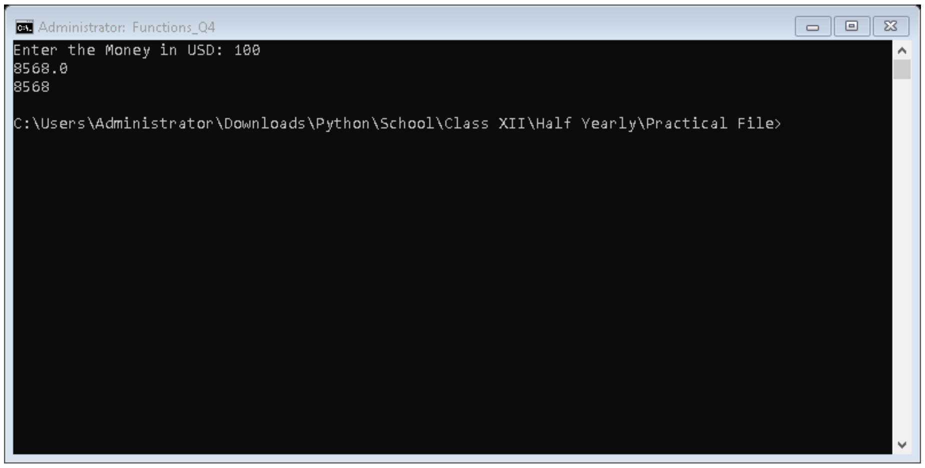
```python
pattern(n)
```



```
Enter a Number: 3
*
* *
* * *

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q4) Write a function that takes amount dollars and dollar-to-rupee conversion price and then returns the amount converted to rupees. Create the function in both void and non-void forms.

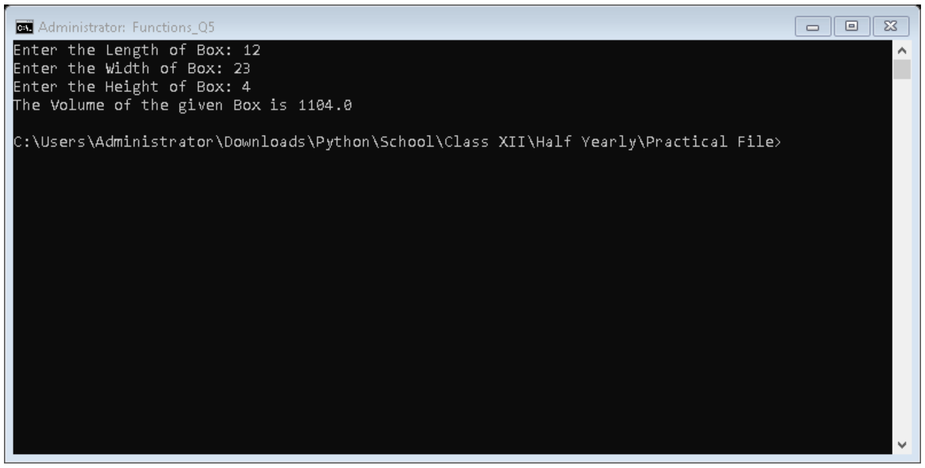
```python
def Void(M):
    print(M*85.68)
def Non_Void(M):
    x=M*85.68
    return int(x)
m=float(input("Enter the Money in USD: "))
Void(m)
print(Non_Void(m))
```



```
Enter the Money in USD: 100
8568.0
8568

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q5) Write a function to calculate the value of a box with appropriate default values for its parameters. Your function should have the following input parameters: Length of box, Width of box, Height of box.

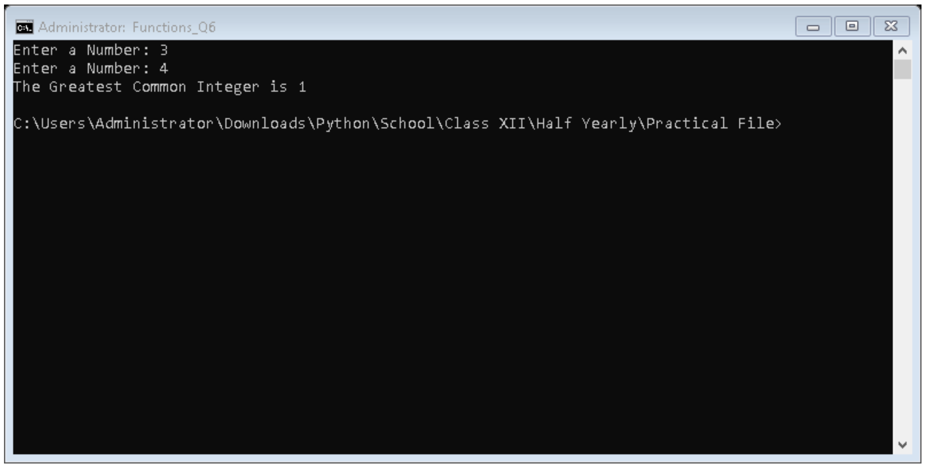
```python
def Volume(L,W,H):
    V=L*W*H
    return V
l=float(input("Enter the Length of Box: "))
w=float(input("Enter the Width of Box: "))
h=float(input("Enter the Height of Box: "))
print("The Volume of the given Box is",Volume(l,w,h))
```



```
Enter the Length of Box: 12
Enter the Width of Box: 23
Enter the Height of Box: 4
The Volume of the given Box is 1104.0

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q6) Write a program to find the greatest common divisor between two numbers.

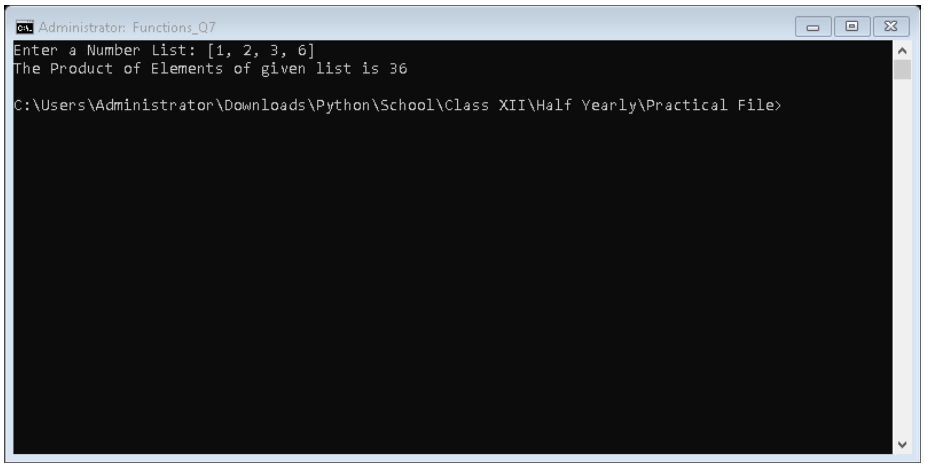
```python
def GCF(N1,N2):
    l1=[]
    l2=[]
    for i in range(1,N1+1):
        if N1%i==0:
            l1.append(i)
    for i in l1:
        if N2%i==0:
            l2.append(i)
    print("The Greatest Common Integer is",max(l2))
n1=int(input("Enter a Number: "))
n2=int(input("Enter a Number: "))
GCF(n1,n2)
```



```
Administrator: Functions_Q6
Enter a Number: 3
Enter a Number: 4
The Greatest Common Integer is 1

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q7) Write a Python function to multiply all the numbers in a list.

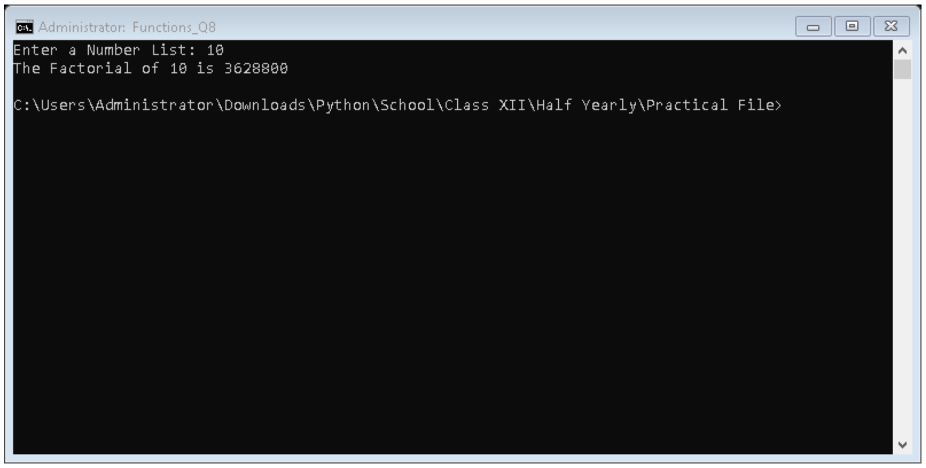
```python
def Multiply(l1):
    p=1
    for i in l1:
        p*=i
    return p
l=eval(input("Enter a Number List: "))
print("The Product of Elements of given list is",Multiply(l))
```



```
Administrator: Functions_Q7
Enter a Number List: [1, 2, 3, 6]
The Product of Elements of given list is 36

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q8) Write a Python function to calculate the factorial of a number (a non-negative integer). The function accepts the number whose factorial is to be calculated as the argument.

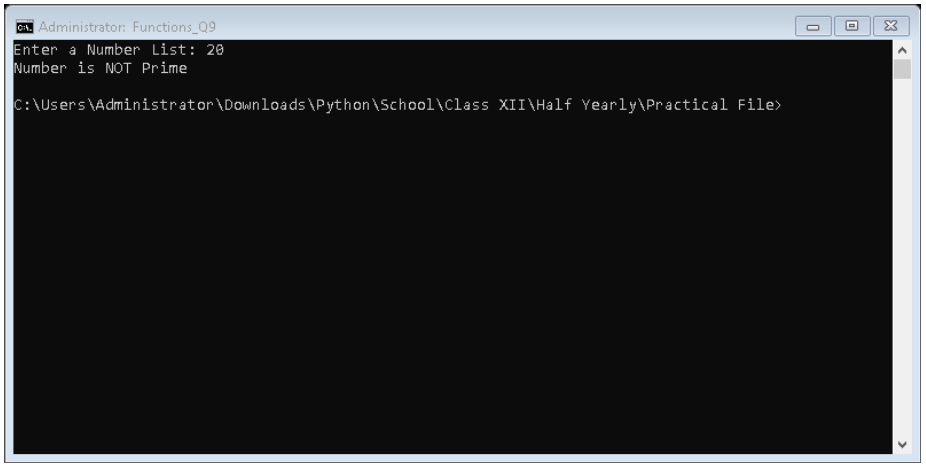
```python
def Multiply(l1):
    p=1
    for i in range(1,l1+1):
        p*=i
    return p
l=eval(input("Enter a Number List: "))
print("The Factorial of",l,"is",Multiply(l))
```



```
Enter a Number List: 10
The Factorial of 10 is 3628800

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q9) Write a Python function that takes a number as a parameter and checks whether the number is prime or not.

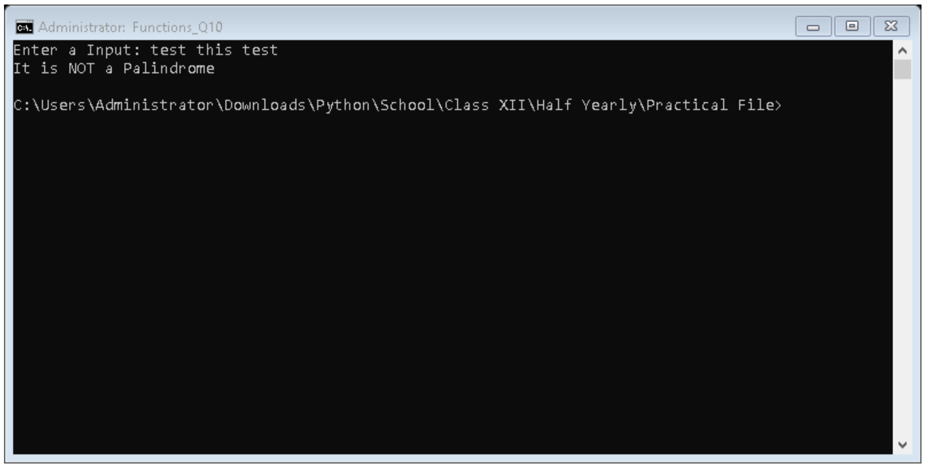
```python
def Prime(l1):
    a=0
    for i in range(2,l1):
        if l1%i==0:
            a=0
            break
        else:
            a=1
    if a==1:
        print("Number is Prime")
    else:
        print("Number is NOT Prime")
l=eval(input("Enter a Number List: "))
Prime(l)
```



```
Enter a Number List: 20
Number is NOT Prime

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q10) Write a Python function that checks whether a passed string is a palindrome or not.

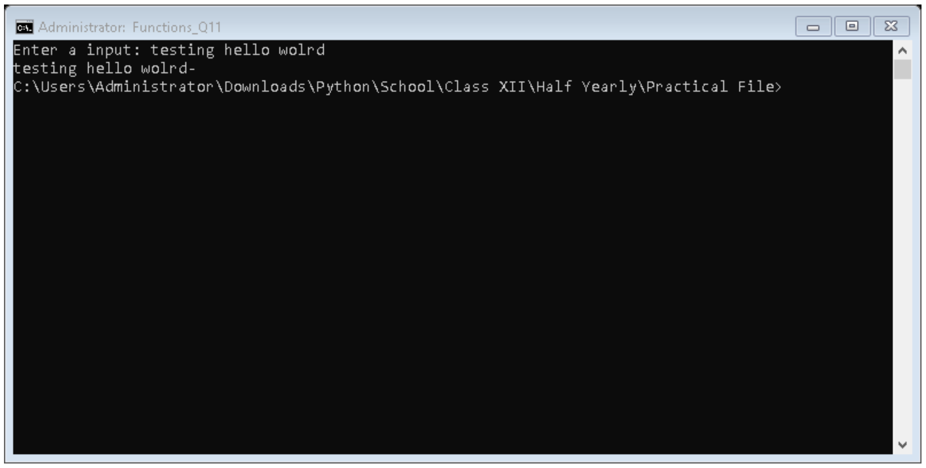
```python
def palindrome(x):
    i=''
    for j in x:
        if j.isalpha()==True:
            i+=j
    if i==i[:-1]:
        print("It is a Palindrome")
    else:
        print("It is NOT a Palindrome")
s=input("Enter a Input: ")
palindrome(s)
```



```
Enter a Input: test this test
It is NOT a Palindrome

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q11) Write a Python program that accepts a hyphen-separated sequence of words as input and prints the words in a hyphen-separated after sorting them alphabetically.

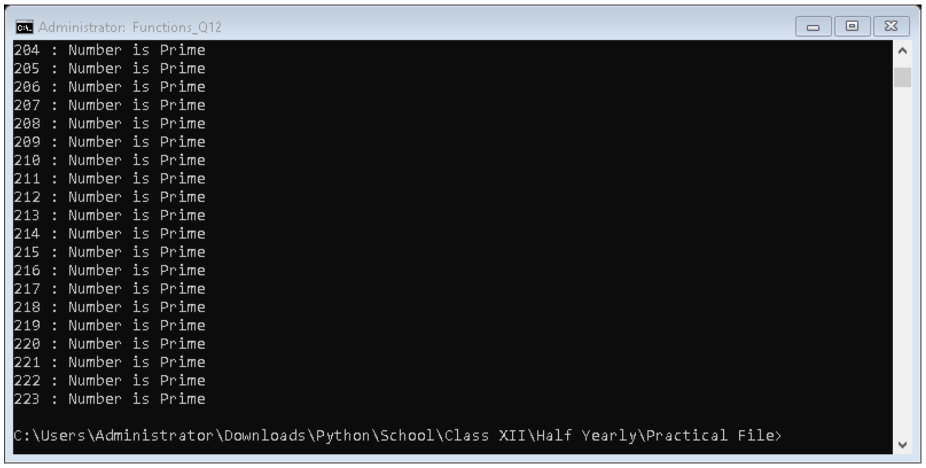
```python
def Sorter(s):
    l1=s.split("-")
    l1.sort()
    for i in l1:
        print(i,end="-")
S=input("Enter a input: ")
Sorter(S)
```



```
Enter a input: testing hello wolrd
testing hello wolrd-
C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q12) Write a method in Python to find and display prime numbers from 2 to N. The value of N should be passed as an argument to the method.

```python
def P_S(N):
    for i in range(2,N+1):
        a=0
        for j in range(2,i):
            if j%i==0:
                a=0
                break
            else:
                a=1
                break
        if a==1:
            print(i,": Number is Prime")
        else:
            print(i,": Number is NOT Prime")
l=eval(input("Enter a Number List: "))
P_S(l)
```

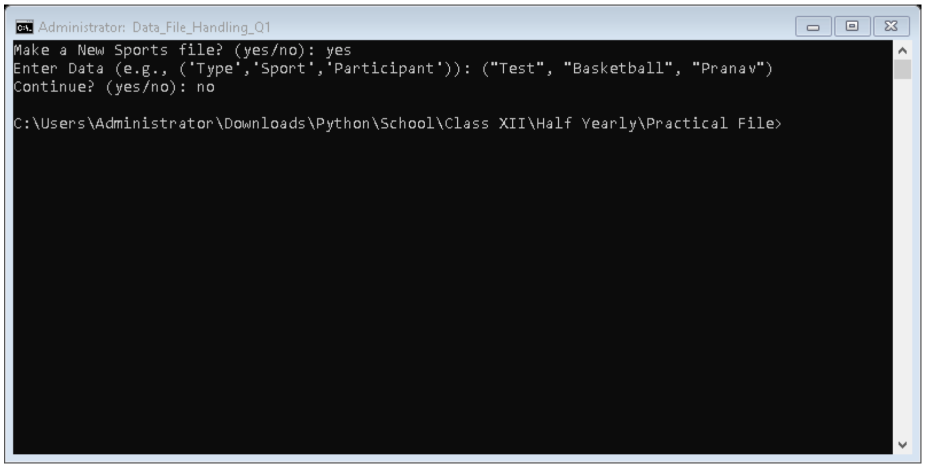


```
204 : Number is Prime
205 : Number is Prime
206 : Number is Prime
207 : Number is Prime
208 : Number is Prime
209 : Number is Prime
210 : Number is Prime
211 : Number is Prime
212 : Number is Prime
213 : Number is Prime
214 : Number is Prime
215 : Number is Prime
216 : Number is Prime
217 : Number is Prime
218 : Number is Prime
219 : Number is Prime
220 : Number is Prime
221 : Number is Prime
222 : Number is Prime
223 : Number is Prime

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

3 Data File Handling

Q1) File 'sports.dat' contains information in the following format: EventName, Participant. Write a function that read contents from file 'sports.dat' and create a file named 'Athletic.dat', copying only those records from 'sports.dat' in which the event name is 'Athletics'.

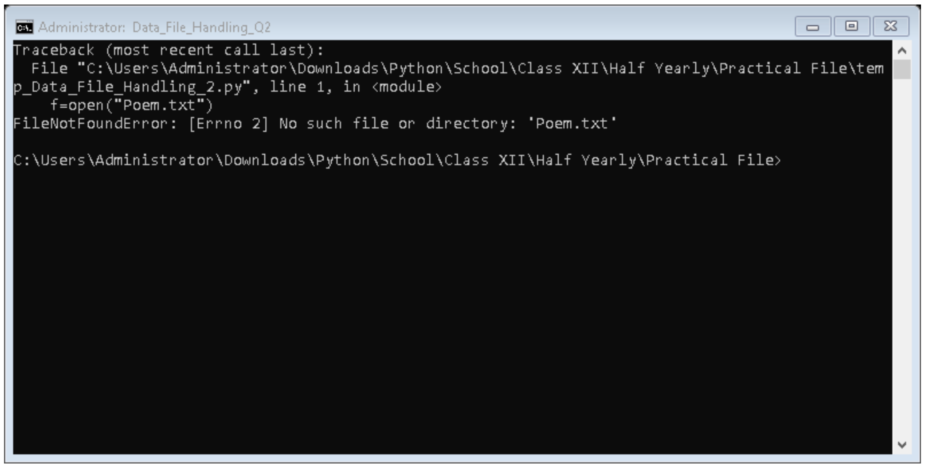
```python
import pickle
ch = input("Make a New Sports file? (yes/no): ")
y = ("yes", "y")
n = ("no", "n")
if ch.lower() in y:
    with open("sports.dat", "wb") as f:
        while True:
            l = eval(input("Enter Data (e.g., ('Type','Sport','
    Participant')): "))
            pickle.dump(l, f)
            c = input("Continue? (yes/no): ")
            if c.lower() in n:
                break
def Athletics():
    with open("sports.dat", "rb") as f:
        with open("Athletics.dat", "wb") as f2:
            try:
                while True:
                    x = pickle.load(f)
                    if isinstance(x, (list, tuple)) and len(x)>0
    and x[0].lower() == "athletics":
                        pickle.dump(x, f2)
            except EOFError:
                print("File Loading Completed!")
    with open("Athletics.dat", "rb") as f:
        try:
            while True:
                print(pickle.load(f))
        except EOFError:
            print("End of File!!")
# Athletics()
```



```
Make a New Sports file? (yes/no): yes
Enter Data (e.g., ('Type','Sport','Participant')): ("Test", "Basketball", "Pranav")
Continue? (yes/no): no

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q2) Write a program to count the words ’to’ and ’the’ present in text file ’Poem.txt’.

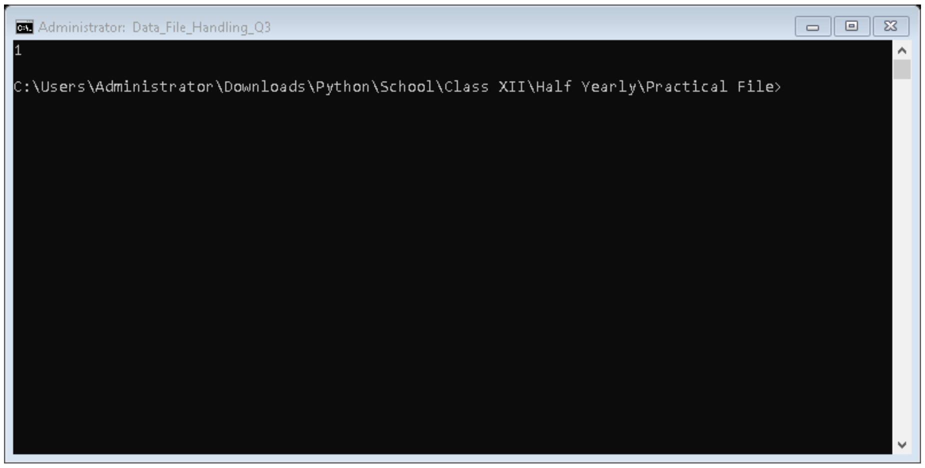
```python
f=open("Poem.txt")
lines=f.read()
f.close()
count=0
line=lines.split()
x=line.index("to")
y=line.index("the")
for i in range(x,y+1):
    count+=1
print(count)
```



```
Traceback (most recent call last):
  File "C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File\tem
p_Data_File_Handling_2.py", line 1, in <module>
    f=open("Poem.txt")
FileNotFoundError: [Errno 2] No such file or directory: 'Poem.txt'

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q3) Write a program to count the number of uppercase alphabets present in text file 'Poem.txt'.

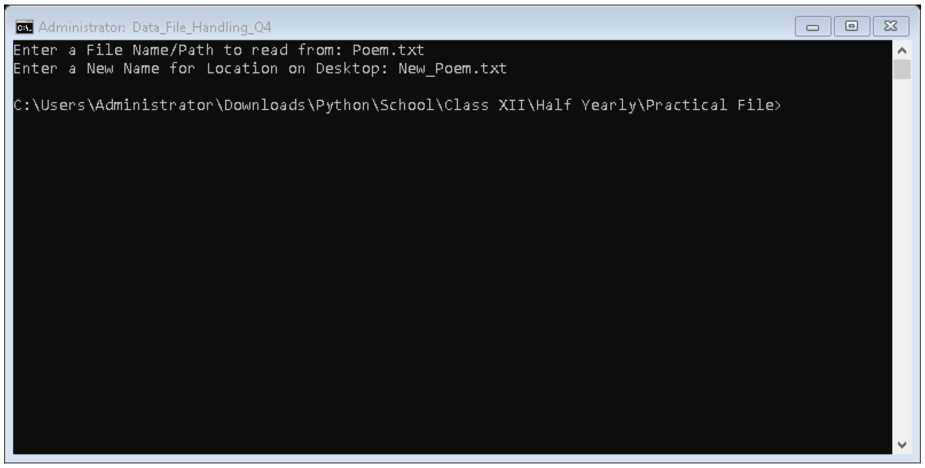
```python
f=open("Poem.txt")
x=f.read()
f.close()
count=0
y="QWERTYUIOPASDFGHJKLZXCVBNM"
for i in x:
    if i in y:
        count+=1
print(count)
```



```
1

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q4) Write a program that copies one file to another and reads the filenames from the user.

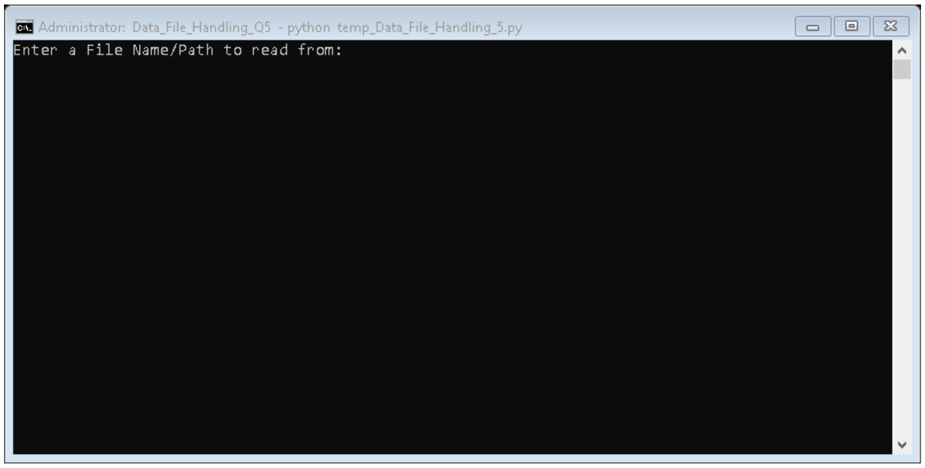
```python
b=input("Enter a File Name/Path to read from: ")
with open(b,"r") as f:
    MyS=f.read()
    f.close()
a=input("Enter a New Name for Location on Desktop: ")
with open(a,"w") as F:
    F.writelines(MyS)
    F.close()
```



```
Enter a File Name/Path to read from: Poem.txt
Enter a New Name for Location on Desktop: New_Poem.txt

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q5) Write a program that appends the contents of one file to another and takes the filenames from the user.

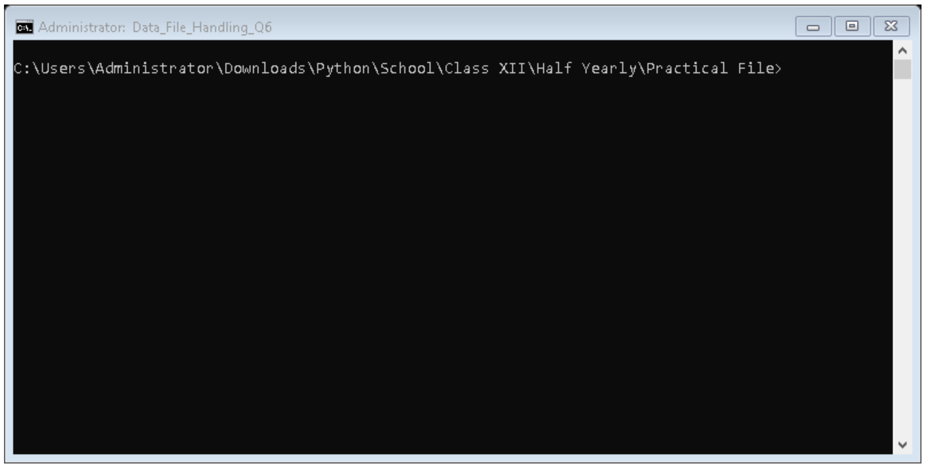
```python
b=input("Enter a File Name/Path to read from: ")
with open(b,"r") as f:
    MyS=f.read()
    f.close()
a=input("Enter a New Name for Location on Desktop: ")
with open(a,"a") as F:
    F.writelines(MyS)
```


Administrator: Data_File_Handling_Q5 - python temp_Data_File_Handling_5.py

```
Enter a File Name/Path to read from:
```

Q6) Write a program that reads characters from the keyboard one by one. All lowercase characters get stored in the file 'LOWER', all uppercase characters get stored in the file 'UPPER' and all the other characters get stored in the file 'OTHERS'.

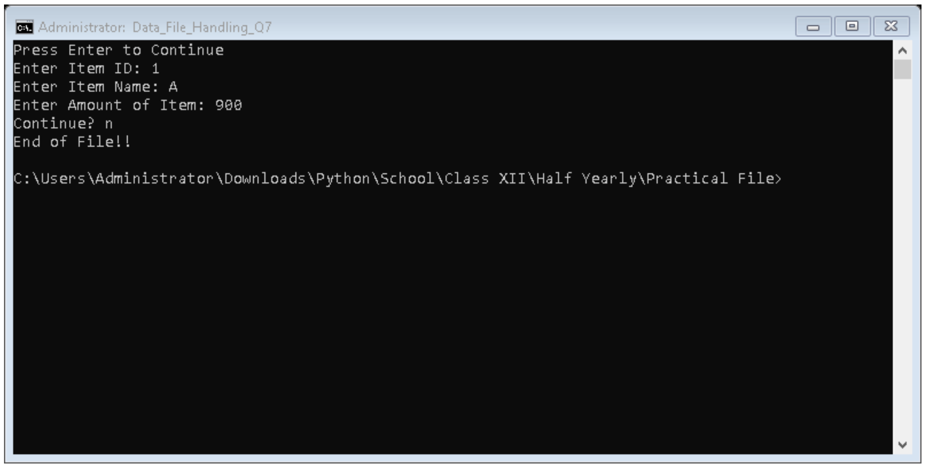
```python
f=open("Poem.txt")
a=f.read()
f.close()
l="asdfghjklqwertyuiopzxcvbnm"
u="QWERTYUIOPASDFGHJKLZXCVBNM"
for i in a:
    if i in l:
        with open("LOWER.txt", "a") as f:
            f.write(i)
            f.close()
    elif i in u:
        with open("UPPER.txt", "a") as f:
            f.write(i)
            f.close()
    elif i!=" ":
        with open("OTHER.txt", "a") as f:
            f.write(i)
            f.close()
```



```
C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q7) Consider binary file 'items.dat' containing records stored in the given format: {item_id: [item_name, amount]}. Write a function, copy_new(), that copies all records whose amount is greater than 1000 from 'items.dat' to 'new_items.dat'.

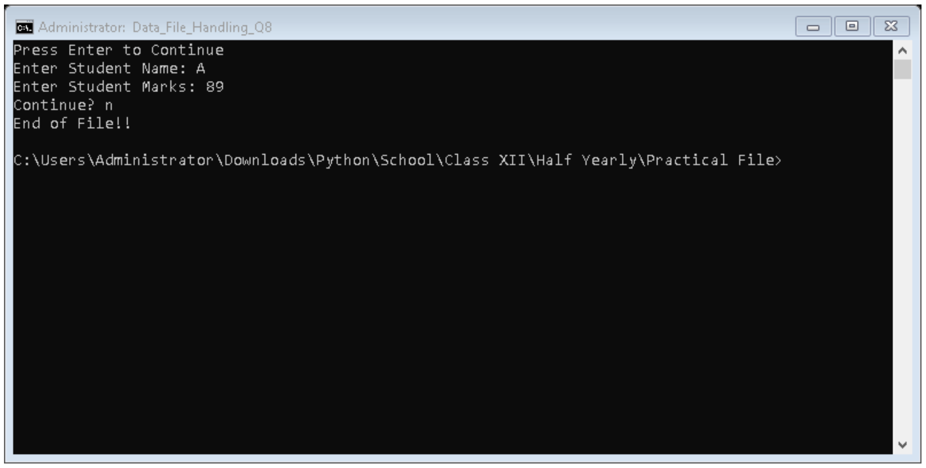
```python
import pickle
with open("items.dat","wb") as f:
    d={}
    c=input("Press Enter to Continue")
    while c.lower()!='n':
        x=input("Enter Item ID: ")
        y=input("Enter Item Name: ")
        z=int(input("Enter Amount of Item: "))
        l=[]
        l.append(y)
        l.append(z)
        d[x]=l
        c=input("Continue? ")
    pickle.dump(d,f)
def copy_new():
    with open("items.dat","rb") as f:
        try:
            d=pickle.load(f)
        except:
            print("End of File!!")
    with open("new_items.dat","wb") as f:
        for i in d:
            if d[i][1]>=1000:
                pickle.dump(d[i],f)
    with open("new_items.dat","rb") as f:
        try:
            while True:
                print(pickle.load(f))
        except:
            print("End of File!!")
copy_new()
```



```
Press Enter to Continue
Enter Item ID: 1
Enter Item Name: A
Enter Amount of Item: 900
Continue? n
End of File!!

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q8) Anant has been asked to display the names of all students who have scored less than 40 for Remedial Classes. Write a user-defined function to display the names of the students from the binary file 'Student.dat' who have less than 40.

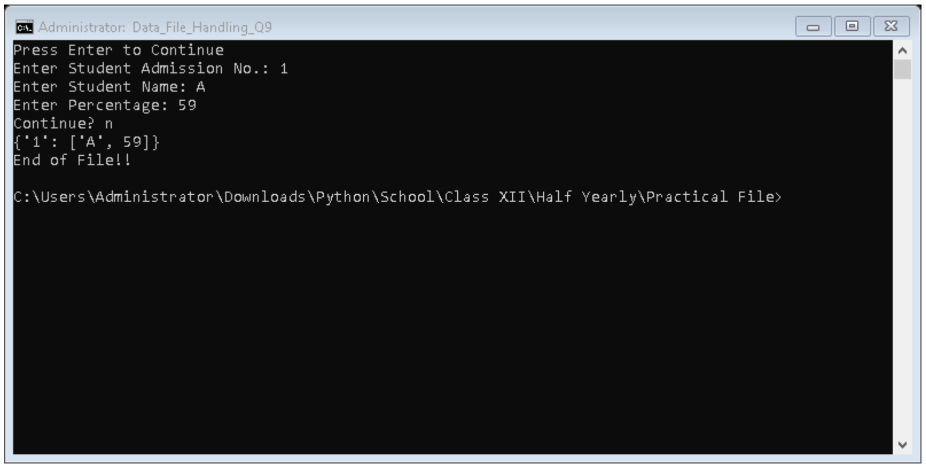
```python
import pickle
with open("Students.dat","wb") as f:
    d={}
    c=input("Press Enter to Continue")
    while c.lower()!='n':
        x=input("Enter Student Name: ")
        l=int(input("Enter Student Marks: "))
        d[x]=l
        c=input("Continue? ")
    pickle.dump(d,f)
def copy_new():
    with open("Students.dat","rb") as f:
        try:
            d=pickle.load(f)
        except:
            print("End of File!!")
    with open("Remedial.dat","wb") as f:
        for i in d:
            if d[i]<=40:
                pickle.dump(i,f)
    with open("Remedial.dat","rb") as f:
        try:
            while True:
                print(pickle.load(f))
        except:
            print("End of File!!")
copy_new()
```



```
Press Enter to Continue
Enter Student Name: A
Enter Student Marks: 89
Continue? n
End of File!!

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q9) Given a binary file, 'STUDENT.dat', containing records of the following type: [S_Admno, S_Name, Percentage] Where these three values are: S_Admno - Admission Number of student (string) S_Name - Name of student (string) Percentage - percentage obtained by student (float) Write a function in Python that would read the contents of the file 'STUDENT.dat' and that would display the details of those students whose percentage is below 65.

```python
import pickle
with open("items.dat","wb") as f:
    d={}
    c=input("Press Enter to Continue")
    while c.lower()!='n':
        x=input("Enter Student Admission No.: ")
        y=input("Enter Student Name: ")
        z=int(input("Enter Percentage: "))
        l=[]
        l.append(y)
        l.append(z)
        d[x]=l
        c=input("Continue? ")
    pickle.dump(d,f)
def copy_new():
    with open("items.dat","rb") as f:
        try:
            d=pickle.load(f)
        except:
            print("End of File!!")
    with open("new_items.dat","wb") as f:
        for i in d:
            if d[i][1]<=65:
                D={}
                D[i]=d[i]
                pickle.dump(D,f)
    with open("new_items.dat","rb") as f:
        try:
            while True:
                print(pickle.load(f))
        except:
            print("End of File!!")
copy_new()
```

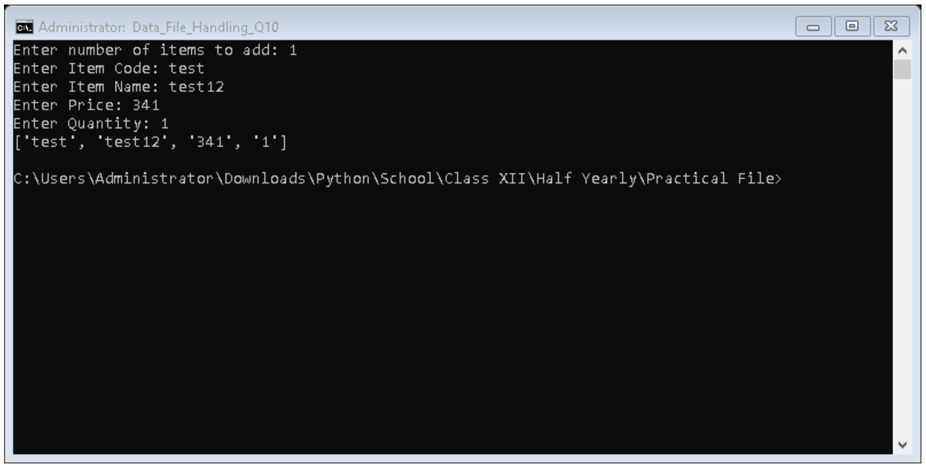


```
Press Enter to Continue
Enter Student Admission No.: 1
Enter Student Name: A
Enter Percentage: 59
Continue? n
{'1': ['A', 59]}
End of File!!

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q10) Create CSV file 'Groceries' to store information of different items existing in a shop. The information is to be stored with respect to each item code, name, price, qty. Write a program to accept the data from the user and store it permanently in the CSV file.

```
import csv
try:
    with open("Groceries.csv", "r") as f_check:
        is_empty = f_check.readline() == ''
except FileNotFoundError:
    is_empty = True
with open("Groceries.csv", "a", newline='') as f:
    writer = csv.writer(f)
    if is_empty:
        writer.writerow(["Item Code", "Item Name", "Price", "
    Quantity"])
    num_rows = int(input("Enter number of items to add: "))
    rows = []
    for _ in range(num_rows):
        code = input("Enter Item Code: ")
        name = input("Enter Item Name: ")
        price = input("Enter Price: ")
        qty = input("Enter Quantity: ")
        rows.append([code, name, price, qty])
    writer.writerows(rows)
```

```python
with open("Groceries.csv", "r", newline='') as f:
    reader = csv.reader(f)
    next(reader)
    for row in reader:
        print(row)
```



```
Enter number of items to add: 1
Enter Item Code: test
Enter Item Name: test12
Enter Price: 341
Enter Quantity: 1
['test', 'test12', '341', '1']

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

4 Stack Operations

Q1) Write a program to implement pop and push functions on a stack containing package name as records.

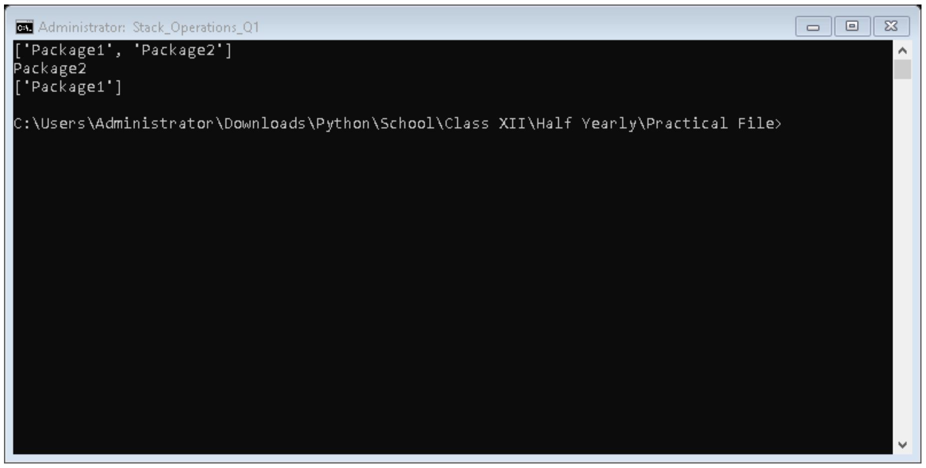
```python
stack = []

def push(item):
    stack.append(item)

def pop():
    if not stack:
        return "Underflow"
    return stack.pop()

# Example usage
push("Package1")
push("Package2")
print(stack)
popped_item = pop()
print(popped_item)
print(stack)
```

Q2) Write a program to sort a stack into ascending order without using another stack.

```python
def sort_stack(stack):
    if stack:
        # Remove the top element
        temp = stack.pop()
        # Sort the remaining stack
        sort_stack(stack)
        # Insert the temp back in sorted position
        sorted_insert(stack, temp)

def sorted_insert(stack, element):
    if not stack or stack[-1] <= element:
        stack.append(element)
    else:
        temp = stack.pop()
        sorted_insert(stack, element)
        stack.append(temp)

# Example usage
stack = [3, 1, 2]
sort_stack(stack)
print(stack)
```

Q1) Write a program to implement pop and push functions on a stack containing package name as records.

```python
stack = []

def push(item):
    stack.append(item)

def pop():
    if not stack:
        return "Underflow"
    return stack.pop()

# Example usage
push("Package1")
push("Package2")
print(stack)
popped_item = pop()
print(popped_item)
print(stack)
```

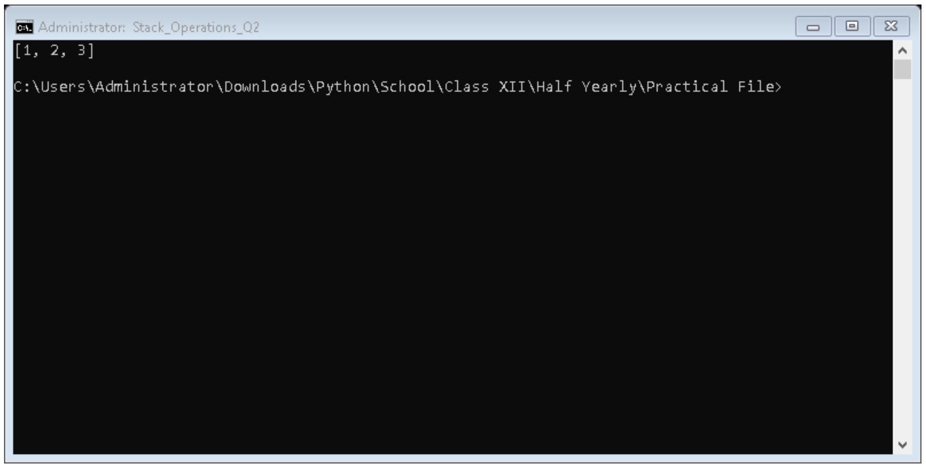


Administrator: Stack_Operations_Q1

```
['Package1', 'Package2']
Package2
['Package1']

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q2) Write a program to sort a stack into ascending order without using another stack.

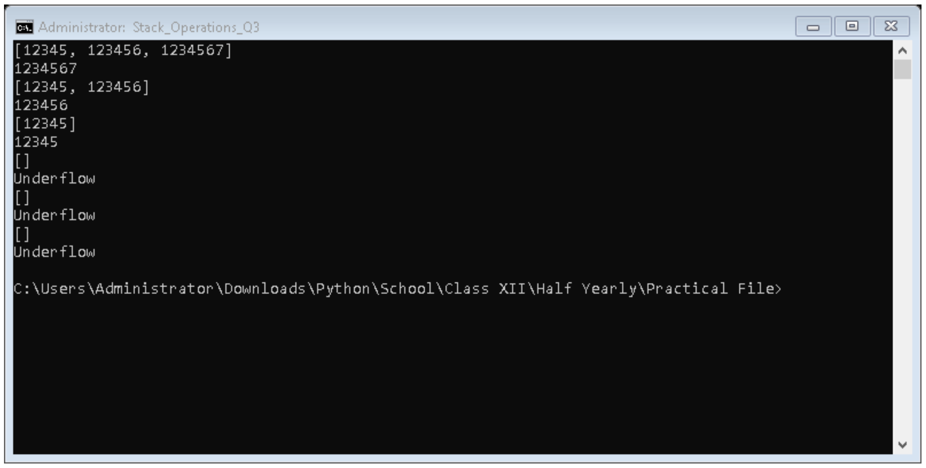
```python
def sort_stack(stack):
    if stack:
        # Remove the top element
        temp = stack.pop()
        # Sort the remaining stack
        sort_stack(stack)
        # Insert the temp back in sorted position
        sorted_insert(stack, temp)

def sorted_insert(stack, element):
    if not stack or stack[-1] <= element:
        stack.append(element)
    else:
        temp = stack.pop()
        sorted_insert(stack, element)
        stack.append(temp)

# Example usage
stack = [3, 1, 2]
sort_stack(stack)
print(stack)
```



```
[1, 2, 3]

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q3) Write a program to implement pop and push operations on a stack. The push operation should add numbers from a list which have 5 digits or more. The pop operation should print underflow if stack is empty.

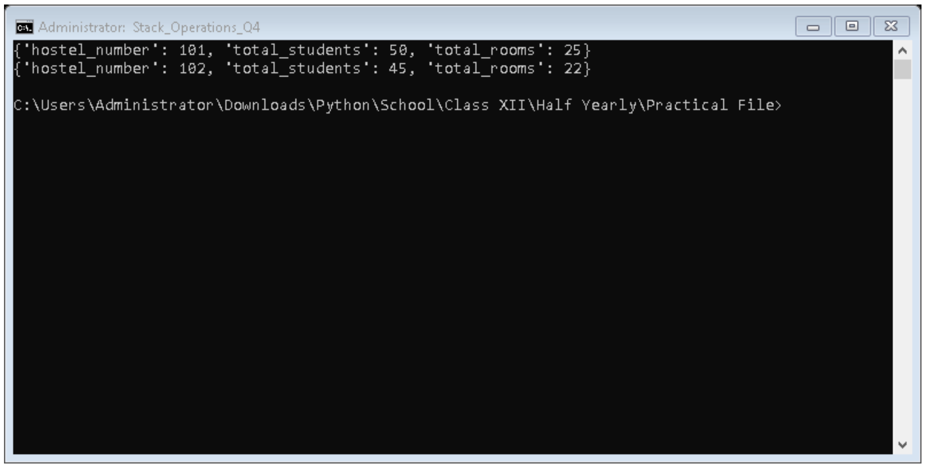
```python
stack = []

def push_from_list(lst):
    for num in lst:
        if len(str(abs(num))) >= 5:  # Check if the number has 5 or more digits
            stack.append(num)

def pop():
    if not stack:
        return "Underflow"
    return stack.pop()

# Example usage
numbers_list = [12345, 1234, 123456, 123, 1234567]
push_from_list(numbers_list)
print(stack)
print(pop())
print(stack)
print(pop())
print(stack)
print(pop())
print(stack)
print(pop())
print(stack)
print(pop())
print(stack)
print(pop())  # Should print "Underflow"
```


Administrator: Stack_Operations_Q3

```
[12345, 123456, 1234567]
1234567
[12345, 123456]
123456
[12345]
12345
[]
Underflow
[]
Underflow
[]
Underflow

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q4) Write a program to implement push and display operations on a stack with hostel number, total students and total rooms as record.

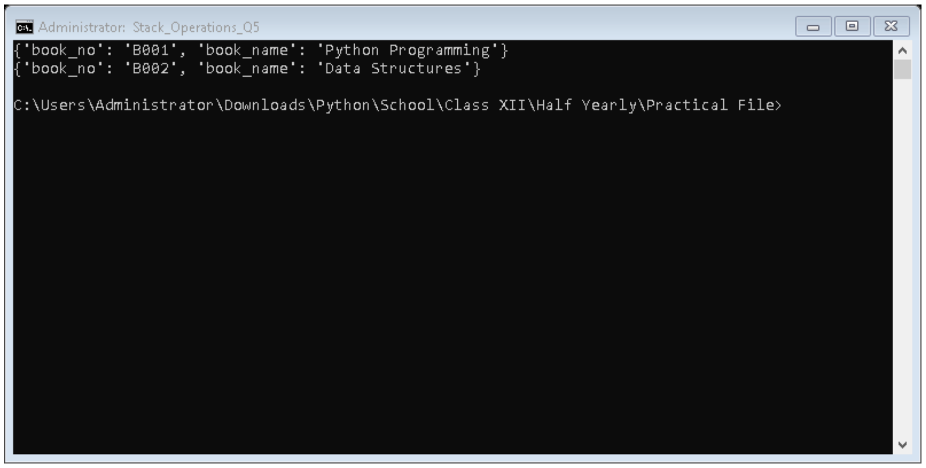
```python
stack = []

def push_hostel_record(hostel_number, total_students, total_rooms):
    record = {
        'hostel_number': hostel_number,
        'total_students': total_students,
        'total_rooms': total_rooms
    }
    stack.append(record)

def display_stack():
    for record in stack:
        print(record)

# Example usage
push_hostel_record(101, 50, 25)
push_hostel_record(102, 45, 22)
display_stack()
```


Administrator: Stack_Operations_Q4

```
{'hostel_number': 101, 'total_students': 50, 'total_rooms': 25}
{'hostel_number': 102, 'total_students': 45, 'total_rooms': 22}

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q5) Write a program to implement push and display operations on a stack which has book number and name as records.

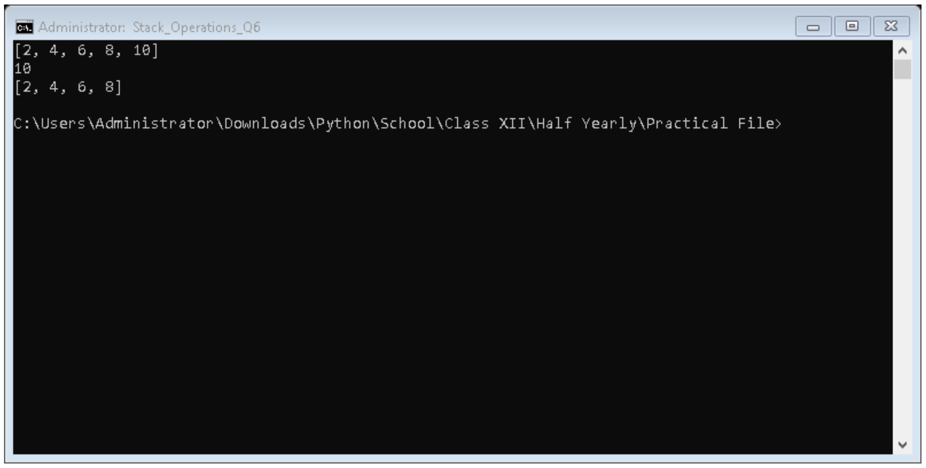
```python
stack = []

def push_book_record(book_no, book_name):
    record = {
        'book_no': book_no,
        'book_name': book_name
    }
    stack.append(record)

def display_stack():
    for record in stack:
        print(record)

# Example usage
push_book_record("B001", "Python Programming")
push_book_record("B002", "Data Structures")
display_stack()
```


Administrator: Stack_Operations_Q5

```
{'book_no': 'B001', 'book_name': 'Python Programming'}
{'book_no': 'B002', 'book_name': 'Data Structures'}

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q6) Write a program to push elements in a stack from a list that are even. Also implement pop function.

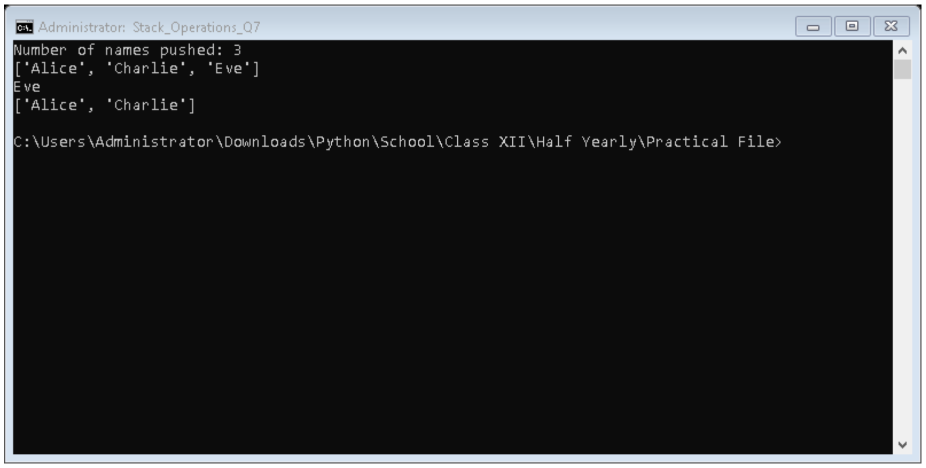
```python
stack = []

def push_evens_from_list(lst):
    for num in lst:
        if num % 2 == 0:
            stack.append(num)

def pop():
    if not stack:
        return "Underflow"
    return stack.pop()

# Example usage
numbers_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
push_evens_from_list(numbers_list)
print(stack)
print(pop())
print(stack)
```



```
[2, 4, 6, 8, 10]
10
[2, 4, 6, 8]

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q7) Write a program to push student names from a dictionary into a stack who have won more than 3 medals. Also it prints the the no of names pushed and overflow if the no of items exceed 15.

```python
stack = []
max_size = 15

def push_students(dictionary):
    count = 0
    for name, medals in dictionary.items():
        if medals > 3:
            if len(stack) >= max_size:
                print("Overflow")
                return
            stack.append(name)
            count += 1
    print(f"Number of names pushed: {count}")

def pop():
    if not stack:
        return "Underflow"
    return stack.pop()

# Example usage
students = {
    "Alice": 4,
    "Bob": 2,
    "Charlie": 5,
    "David": 3,
    "Eve": 6
}
push_students(students)
print(stack)
print(pop())
print(stack)
```



```
Number of names pushed: 3
['Alice', 'Charlie', 'Eve']
Eve
['Alice', 'Charlie']

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

5 Relational Databases and SQL

Q10) Give the terms for each of the following:

```
(a) Database or File
(b) Table or Relation
(c) Primary Key
(d) NULL
(e) Candidate Key or Alternate Key
(f) RDBMS (Relational Database Management System)
```

Q11) An organization wants to create two tables EMP and DEPENDENT

EMPLOYEE(AadhaarNumber, Name, Address, Department, EmployeeID)
DEPENDENT(EmployeeID, DependentName, Relationship)

```
(a) AadhaarNumber and EmployeeID

(b) Tables: EMPLOYEE and DEPENDENT
    Key: EmployeeID (Foreign Key in DEPENDENT)

(c) Degree of EMPLOYEE: 5
    Degree of DEPENDENT: 3
```

Q12) What is a data type? Name some data types available in MySQL.

```
Data type specifies the type of data that can be stored in a column
    .

MySQL Data Types:
```

```
INT, FLOAT, DOUBLE, CHAR, VARCHAR, TEXT, DATE,
DATETIME, TIMESTAMP, BOOLEAN, DECIMAL, BLOB
```

Q13) Differentiate between char and varchar data types.

```
CHAR:
- Fixed length
- Uses full allocated space
- Faster for fixed-length data
- Pads with spaces

VARCHAR:
- Variable length
- Uses only required space
- More efficient for variable-length data
- No padding
```

Q14) Which operator concatenates two strings in a query result?

```
CONCAT() function or || operator
```

Q15) How would you calculate 13*15 in SQL?

```
SELECT 13 * 15;
```

Q16) Which keywords eliminate redundant data from a query?

```
DISTINCT
```

Q17) What is the significance of GROUP BY clause in an SQL query?

```
GROUP BY clause groups rows with same values in specified columns.
It is used with aggregate functions like COUNT, SUM, AVG, MAX, MIN.
```

Q18) What is the difference between WHERE and HAVING clause?

```
WHERE:
- Filters rows before grouping
- Cannot use aggregate functions
- Applied to individual rows
```

```

HAVING:
- Filters groups after grouping
- Can use aggregate functions
- Applied to grouped results
```

Q19) Write SQL queries based on table PRODUCT

```
(a) SELECT * FROM PRODUCT WHERE prod_id > 100;

(b) SELECT * FROM PRODUCT WHERE prod_name = 'Almirah';

(c) SELECT * FROM PRODUCT WHERE price BETWEEN 200 AND 500;

(d) SELECT prod_name FROM PRODUCT
    WHERE price < (SELECT AVG(price) FROM PRODUCT);

(e) SELECT COUNT(*) FROM PRODUCT;
```

Q20) Help Ms. Veda add the Category column

```
ALTER TABLE game ADD Category VARCHAR(20);
```

Q21) Define the following terms:

```
(a) Database: Organized collection of related data

(b) Data Inconsistency: Same data stored differently in
    multiple places causing conflicts

(c) Primary Key: Attribute that uniquely identifies each
    record in a table

(d) Candidate Key: Attribute that can be used as primary key

(e) Foreign Key: Attribute that references primary key of
    another table
```

Q22) Differentiate between Primary key and Unique constraints.

```
Primary Key:
- Only one per table
- Cannot contain NULL values
- Creates clustered index

Unique Constraint:
- Multiple allowed per table
- Can contain one NULL value
- Creates non-clustered index
```

Q23) Write SQL commands to:

```
(i) SHOW TABLES FROM Exam;

(ii) DESCRIBE Term1;
```

Q24-25) Consider the following EMP and DEPT tables:

EmpNo	EmpName	City	Designation	DOJ	Sal	Comm	DeptID
8369	SMITH	Mumbai	CLERK	1990-12-18	800.00	NULL	20
8499	ANYA	Varanasi	SALESMAN	1991-02-20	1600.00	300.00	30
8521	SETH	Jaipur	SALESMAN	1991-02-22	1250.00	500.00	30
8566	MAHADEVAN	Delhi	MANAGER	1991-04-02	2985.00	NULL	20

DeptID	DeptName	MgrID	Location
10	SALES	8566	Mumbai
20	PERSONNEL	9698	Delhi
30	ACCOUNTS	4578	Delhi
40	RESEARCH	8839	Bengaluru

```
(a) SELECT MIN(Sal), MAX(Sal), AVG(Sal) FROM EMP
    WHERE Designation = 'MANAGER';

(b) SELECT COUNT(*) FROM EMP WHERE Designation = 'CLERK';

(c) SELECT Designation, EmpName, Sal, DOJ FROM EMP
    ORDER BY Designation;

(d) SELECT COUNT(*) FROM EMP WHERE Comm IS NULL;

(e) SELECT DeptID, AVG(Sal) FROM EMP
    GROUP BY DeptID HAVING AVG(Sal) > 2000;

(f) SELECT DeptID, COUNT(*) FROM EMP GROUP BY DeptID;

(g) SELECT DeptID, MAX(Sal) FROM EMP GROUP BY DeptID;

(h) SELECT E.EmpName, E.Designation, D.DeptName
    FROM EMP E, DEPT D WHERE E.DeptID = D.DeptID;

(i) SELECT COUNT(*) FROM EMP E, DEPT D
    WHERE E.DeptID = D.DeptID AND D.DeptName = 'ACCOUNTS';
```

Q26) Consider the table PRODUCTS:

PCODE	PNAME	COMPANY	PRICE	STOCK	MANUFACTURE	WARRANTY
P001	TV	BPL	10000	200	2018-01-12	3
P002	TV	SONY	12000	150	2017-03-23	4
P003	PC	LENOVO	39000	100	2018-04-09	2
P004	PC	COMPAQ	38000	120	2019-06-20	2
P005	HANDYCAM	SONY	18000	250	2017-03-23	3

```
(a) SELECT * FROM PRODUCTS WHERE PNAME = 'PC' AND STOCK > 110;

(b) SELECT DISTINCT COMPANY FROM PRODUCTS WHERE WARRANTY > 2;

(c) SELECT SUM(PRICE * STOCK) FROM PRODUCTS WHERE COMPANY = 'BPL';

(d) SELECT COMPANY, COUNT(*) FROM PRODUCTS GROUP BY COMPANY;

(e) SELECT COUNT(*) FROM PRODUCTS
    WHERE DATE_ADD(MANUFACTURE, INTERVAL WARRANTY YEAR) = '
    2020-11-20';

(f) SELECT PNAME FROM PRODUCTS
    WHERE DATE_ADD(MANUFACTURE, INTERVAL WARRANTY YEAR) >= CURDATE
    ();

(g) 3

(h) 12000
```

Q27) Write SQL commands based on PROJECTS and BOOKS tables:

P_Id	Pname	Language	Startdate	Enddate
P001	School Management System	Python	2023-01-12	2023-04-03
P002	Hotel Management System	C++	2022-02-01	2023-02-02
P003	Blood Bank	Python	2023-02-11	2023-03-02
P004	Payroll Management System	Python	2023-03-12	2023-06-02

Book_ID	Book_name	Author_name	Publishers	Price	Type	Qty
K0001	Let us C	Y. Kanetkar	EPB	450	Prog	15
P0001	Computer Networks	B. Agarwal	FIRST PUBL	755	Comp	24
M0001	Mastering C++	K.R. Venugopal	EPB	165	Prog	60
N0002	VC++ advance	P. Purohit	TDH	350	Prog	45
K0002	Programming with Python	Sanjeev	FIRST PUBL	350	Prog	30

```
SELECT Pname FROM PROJECTS WHERE Language = 'Python';

SELECT Book_name FROM BOOKS WHERE Price BETWEEN 200 AND 500;

SELECT Book_name, Publishers FROM BOOKS WHERE Type = 'Prog';

SELECT COUNT(*) FROM BOOKS WHERE Publishers = 'EPB';
```

Q28) What are DDL and DML?

```
DDL (Data Definition Language):
Commands that define database structure
Examples: CREATE, ALTER, DROP, TRUNCATE

DML (Data Manipulation Language):
Commands that manipulate data in database
Examples: SELECT, INSERT, UPDATE, DELETE
```

Q29) Differentiate between primary key and candidate key.

```
Primary Key:
- Selected candidate key for unique identification
- Only one per table
- Cannot be NULL

Candidate Key:
- Any attribute that can uniquely identify records
- Multiple candidate keys possible
- One becomes primary key, others are alternate keys
```

Q30) What is Cardinality and Degree of a relation?

```
Cardinality: Number of rows (tuples) in a table

Degree: Number of columns (attributes) in a table
```

Q31) Differentiate between DDL and DML with commands.

```
DDL (Data Definition Language):
- Defines structure
- Auto-commit
Examples: CREATE, ALTER

DML (Data Manipulation Language):
- Manipulates data
- Requires commit
Examples: INSERT, UPDATE
```

Q32-33) Write SQL Commands based on GRADUATE table:

S.No.	NAME	STIPEND	SUBJECT	AVERAGE	RANK
1	KARAN	400	PHYSICS	68	1
2	RAJ	450	CHEMISTRY	68	1
3	DEEP	300	MATHS	62	2
4	DIVYA	350	CHEMISTRY	63	1
5	GAURAV	500	PHYSICS	70	1
6	MANAV	400	CHEMISTRY	55	2
7	VARUN	250	MATHS	64	1
8	LIZA	450	COMPUTER	68	1
9	PUJA	500	PHYSICS	62	2
10	NISHA	300	COMPUTER	57	2

```sql
SELECT * FROM GRADUATE WHERE SUBJECT = 'PHYSICS';

SELECT NAME, STIPEND FROM GRADUATE WHERE AVERAGE >= 65;

SELECT SUBJECT, AVG(STIPEND) FROM GRADUATE GROUP BY SUBJECT;

SELECT COUNT(*) FROM GRADUATE WHERE RANK = 1;

SELECT MAX(AVERAGE) FROM GRADUATE WHERE SUBJECT = 'CHEMISTRY';

SELECT NAME FROM GRADUATE WHERE STIPEND BETWEEN 300 AND 450;

SELECT DISTINCT SUBJECT FROM GRADUATE;
```

Q34) Answer the following:

```
(a) Candidate Key: Any attribute that can uniquely identify records
    Alternate Key: Candidate keys not selected as primary key

(b) Degree: 5 columns
    Cardinality: 10 rows

(c) CREATE TABLE STUDENT (
    ROLLNO INTEGER(4) PRIMARY KEY,
    SNAME VARCHAR(25) NOT NULL,
    GENDER CHAR(1) NOT NULL,
    DOB DATE NOT NULL,
    FEES INTEGER(4) NOT NULL,
    HOBBY VARCHAR(15)
);
```

Q35) Write SQL queries based on PRODUCT and CLIENT tables:

P_ID	ProductName	Manufacturer	Price	Discount
TP01	Talcum Powder	LAK	40	NULL
FW05	Face Wash	ABC	45	5
BS01	Bath Soap	ABC	55	NULL
SH06	Shampoo	XYZ	120	10
FW12	Face Wash	XYZ	95	NULL

C_ID	ClientName	City	P_ID
01	Cosmetic Shop	Delhi	TP01
02	Total Health	Mumbai	FW05
03	Live Life	Delhi	BS01
04	Pretty Woman	Delhi	SH06
05	Dreams	Delhi	FW12

```sql
(a) SELECT ProductName, Price FROM PRODUCT
    WHERE Price BETWEEN 50 AND 150;

(b) SELECT * FROM PRODUCT
    WHERE Manufacturer IN ('XYZ', 'ABC');

(c) SELECT ProductName, Manufacturer, Price FROM PRODUCT
    WHERE Discount IS NULL;
```

```
(d) SELECT ProductName, Price FROM PRODUCT
    WHERE ProductName LIKE '%h';

(e) SELECT C.ClientName, C.City, C.P_ID, P.ProductName
    FROM CLIENT C, PRODUCT P
    WHERE C.P_ID = P.P_ID AND C.City = 'Delhi';

(f) P_ID is used as Foreign Key in CLIENT table
```

Q36) Consider the table ORDERS and write output:

ORDNO	ITEM	QTY	RATE	ORDATE
1001	RICE	23	120	2023-09-10
1002	PULSES	13	120	2023-10-18
1003	RICE	25	110	2023-11-17
1004	WHEAT	28	65	2023-12-25
1005	PULSES	16	110	2024-01-15
1006	WHEAT	27	55	2024-04-15
1007	WHEAT	25	60	2024-04-30

```
(i) ITEM      SUM(QTY)
    RICE      48
    PULSES    29
    WHEAT     80

(ii) ITEM      QTY
     RICE      25
     WHEAT     28

(iii) ORDNO    ORDATE
      1004     2023-12-25
      1007     2024-04-30
```

Q37) Answer based on HOSPITAL table:

S.No.	Name	Age	Department	Dateofadm	Charges	Sex
1	Arpit	62	Surgery	1998-01-21	300	M
2	Zareena	22	ENT	1997-02-12	200	F
3	Kareem	32	Orthopaedic	1998-02-19	250	M
4	Arun	12	Surgery	1998-01-19	300	M
5	Zubin	30	ENT	1998-01-12	200	M
6	Ketaki	16	ENT	1998-02-24	250	F
7	Ankit	29	Cardiology	1998-02-20	250	M
8	Zareen	45	Gynaecology	1998-02-22	800	F
9	Kush	19	Cardiology	1998-01-13	300	M
10	Shilpa	23	Nuclear Medicine	1998-02-21	400	F

```
(a) SELECT Name FROM HOSPITAL WHERE Dateofadm > '1998-01-15';

```

```
(b) SELECT Name FROM HOSPITAL
    WHERE Sex = 'F' AND Department = 'ENT';

(c) SELECT Name, Dateofadm FROM HOSPITAL ORDER BY Dateofadm;

(d) SELECT Name, Charges, Age FROM HOSPITAL WHERE Sex = 'F';

(e) (i) 5
    (ii) 16
```

Q38) A department store MyStore maintains inventory:

ItemNo	ItemName	SCode	Quantity
2005	Sharpener Classic	23	60
2003	Ball Pen 0.25	22	50
2002	Gel Pen Premium	21	150
2006	Gel Pen Classic	21	250
2001	Eraser Small	22	220
2004	Eraser Big	22	110
2009	Ball Pen 0.5	21	180

```
(a) ItemNo

(b) Degree: 4
    Cardinality: 7

(c) INSERT INTO STORE (ItemNo, ItemName, SCode)
    VALUES (2010, 'Note Book', 25);

(d) (ii) DROP TABLE STORE;

(e) DESCRIBE STORE;
```

Q39) Consider tables Admin and Transport:

S_id	S_name	Address	S-type
S001	Sandhya	Rohini	Day Boarder
S002	Vedanshi	Rohtak	Day Scholar
S003	Vibhu	Raj Nagar	NULL
S004	Atharva	Rampur	Day Boarder

S_id	S_name	S-type
S002	TSS10	Sarai Kale Khan
S004	TSS12	Sainik Vihar
S005	TSS10	Kamla Nagar

```
(i) SELECT A.S_name, T.S_type
    FROM Admin A, Transport T
    WHERE A.S_id = T.S_id;

(ii) SELECT COUNT(*) FROM Admin WHERE S_type IS NULL;
```

```

(iii) SELECT * FROM Admin WHERE S_name LIKE 'V%';

(iv) SELECT S_id, Address FROM Admin ORDER BY S_name;
```