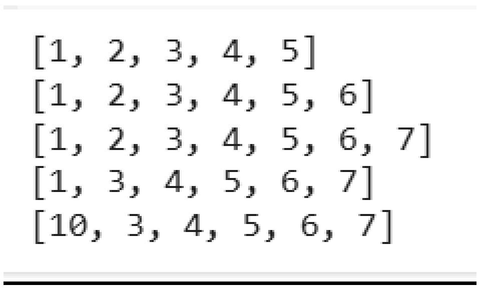
EX.NO: 06	LIST AND TUPLE
DATE:	

PROGRAM 1:

Write a program to create a list and perform the following operations: append, insert, delete, and update an element.

```
a=[1,2,3,4,5]
print(a)
a.append(6)
print(a)
a.insert(6,7)
print(a)
a.pop(1)
print(a)
a[0]=10
print(a)
```

OUTPUT:



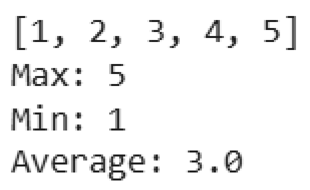
```
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5, 6]
[1, 2, 3, 4, 5, 6, 7]
[1, 3, 4, 5, 6, 7]
[10, 3, 4, 5, 6, 7]
```

PROGRAM 2:

Write a program to create a list of integers and print the largest, smallest, and average of the list elements.

```
a = [1, 2, 3, 4, 5]
print(a)
print("Max:", max(a))
print("Min:", min(a))
average = sum(a) / len(a)
print("Average:", average)
```

OUTPUT:



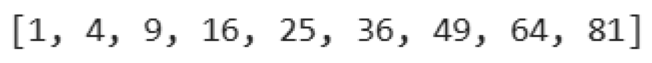
```
[1, 2, 3, 4, 5]
Max: 5
Min: 1
Average: 3.0
```

PROGRAM 3:

Write a program to generate a list of squares of the first 10 natural numbers using list comprehension

```
s=[x**2 for x in range(1,10)]
print(s)
```

OUTPUT:



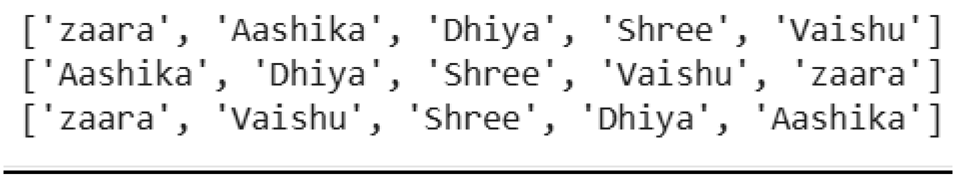
```
[1, 4, 9, 16, 25, 36, 49, 64, 81]
```

PROGRAM 4:

Write a program to create a list of student names and sort them in ascending and descending order

```
students_names=['zaara','Aashika','Dhiya','Shree','Vaishu']
print(students_names)
students_names.sort()
print(students_names)
students_names.sort(reverse=True)
print(students_names)
```

OUTPUT:



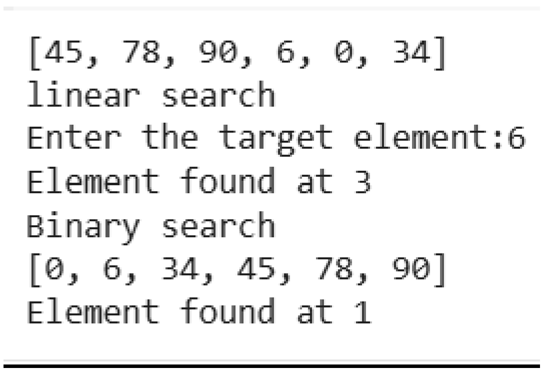
```
['zaara', 'Aashika', 'Dhiya', 'Shree', 'Vaishu']
['Aashika', 'Dhiya', 'Shree', 'Vaishu', 'zaara']
['zaara', 'Vaishu', 'Shree', 'Dhiya', 'Aashika']
```

PROGRAM 5:

Write a program to perform linear search and binary search on a list of numbers

```
list=[45,78,90,6,0,34]
print(list)
print("linear search")
target=int(input("Enter the target element:"))
if(target in list):
  print("Element found at",list.index(target))
else:
  print("Element not found")
print("Binary search")
sorted_list=sorted(list)
print(sorted_list)
low=0
high=len(sorted_list)-1

while low<=high:
  mid=(low+high)//2
  if sorted_list[mid]==target:
    print("Element found at",mid)
    break
  elif sorted_list[mid]<target:
    low=mid+1
  else:
    high=mid-1
else:
  print("Element not found")
```

OUTPUT:



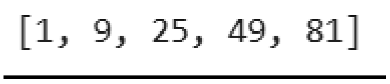
```
[45, 78, 90, 6, 0, 34]
linear search
Enter the target element:6
Element found at 3
Binary search
[0, 6, 34, 45, 78, 90]
Element found at 1
```

PROGRAM 6:

Given a list of numbers, create a new list containing the squares of only the odd numbers from the original list. (Use List comprehension)

```
list=[1,2,3,4,5,6,7,8,9,10]
new_list=[x**2 for x in range(len(list)) if x%2!=0]
print(new_list)
```

OUTPUT:



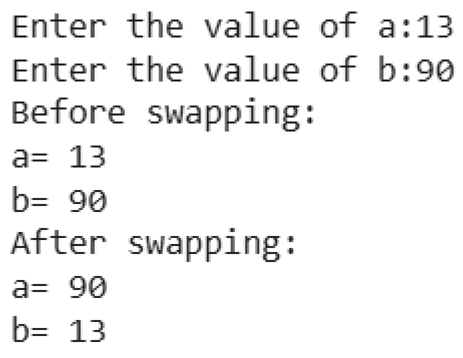
```
[1, 9, 25, 49, 81]
```

PROGRAM 7:

Write a program to swap two values using tuple unpacking.

```
a=int(input("Enter the value of a:"))
b=int(input("Enter the value of b:"))
print("Before swapping:")
print("a=",a)
print("b=",b)
a,b=b,a
print("After swapping:")
print("a=",a)
print("b=",b)
```

OUTPUT:



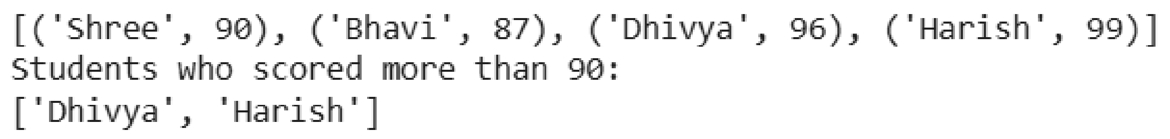
```
Enter the value of a:13
Enter the value of b:90
Before swapping:
a= 13
b= 90
After swapping:
a= 90
b= 13
```

PROGRAM 8:

Write a program to create a list of tuples containing student names and their marks, and print the names of students who scored more than 90.

```
students=[("Shree",90), ("Bhavi",87), ("Dhivya",96), ("Harish", 99)]
print(students)
high_score=[name for name, marks in students if marks>90]
print("Students who scored more than 90:")
print(high_score)
```

OUTPUT:



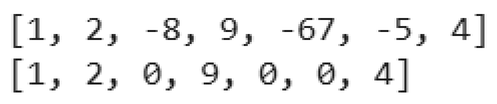
```
[('Shree', 90), ('Bhavi', 87), ('Dhivya', 96), ('Harish', 99)]
Students who scored more than 90:
['Dhivya', 'Harish']
```

PROGRAM 9:

A sensor returns a list of readings. Negative readings are invalid and should be replaced by zero and Use list comprehension to replace all negative numbers in a list with 0

```
readings=[1,2,-8,9,-67,-5,4]
print(readings)
new_readings=[x if x>=0 else 0 for x in readings]
print(new_readings)
```

OUTPUT:



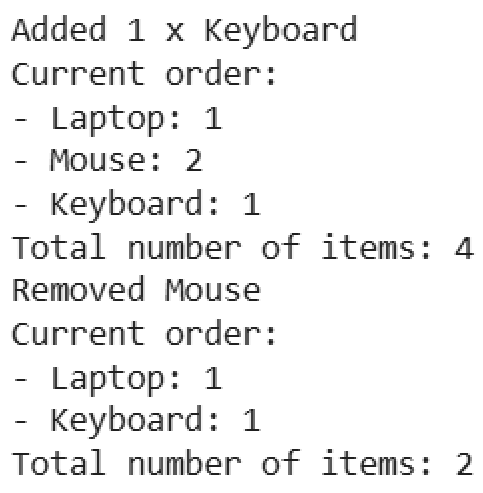
```
[1, 2, -8, 9, -67, -5, 4]
[1, 2, 0, 9, 0, 0, 4]
```

PROGRAM 10:

Create a system to process customer orders. Each order can contain multiple items (represented as strings). The system should allow adding items to an order, removing items, displaying the current order, and calculating the total number of items in the order. You could extend this by associating quantities with each item (e.g., a list of tuples: [("Laptop", 1), ("Mouse", 2)]).

```
order = [("Laptop", 1), ("Mouse", 2)]

def add_item(item, quantity):
    order.append((item, quantity))
    print(f"Added {quantity} x {item}")

def remove_item(item):
    for i in order:
        if i[0] == item:
            order.remove(i)
            print(f"Removed {item}")
            return
    print(f"{item} not found in order")

def show_order():
    if not order:
        print("Order is empty.")
    else:
        print("Current order:")
        for item in order:
            print(f"- {item[0]}: {item[1]}")

def total_items():
    total = 0
    for item in order:
        total += item[1]
    print("Total number of items:", total)

add_item("Keyboard", 1)
show_order()
total_items()
remove_item("Mouse")
show_order()
total_items()
```

OUTPUT:



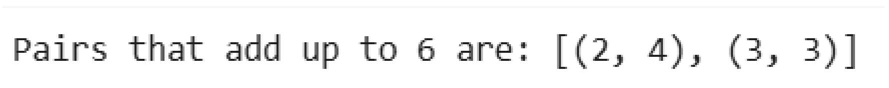
```
Added 1 x Keyboard
Current order:
- Laptop: 1
- Mouse: 2
- Keyboard: 1
Total number of items: 4
Removed Mouse
Current order:
- Laptop: 1
- Keyboard: 1
Total number of items: 2
```

PROGRAM 11:

Given a list of numbers and a target sum, find all unique pairs of numbers in the list that add up to the target sum

```
nums = [1, 2, 3, 4, 3]
target = 6
pairs = []

for i in range(len(nums)):
    for j in range(i + 1, len(nums)):
        if nums[i] + nums[j] == target:
            if (nums[j], nums[i]) not in pairs and (nums[i], nums[j]) not in pairs:
                pairs.append((nums[i], nums[j]))

print("Pairs that add up to", target, "are:", pairs)
```

OUTPUT:



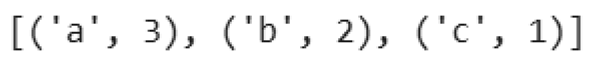
```
Pairs that add up to 6 are: [(2, 4), (3, 3)]
```

PROGRAM 12:

Implement a function that performs Run-Length Encoding on a list of characters. RLE compresses data by replacing consecutive identical characters with the character and its count. For example, ["a", "a", "a", "b", "b", "c"] would be encoded as [("a", 3), ("b", 2), ("c", 1)].

```
def run_length_encoding(chars):
    result = []
    count = 1

    for i in range(1, len(chars)):
        if chars[i] == chars[i - 1]:
            count += 1
        else:
            result.append((chars[i - 1], count))
            count = 1

    result.append((chars[-1], count))
    return result
chars = ["a", "a", "a", "b", "b", "c"]
print(run_length_encoding(chars))
```

OUTPUT:



```
[('a', 3), ('b', 2), ('c', 1)]
```

PROGRAM 13:

1. Given a matrix and need to transpose it and Use nested list comprehension to transpose a matrix.

   Input:

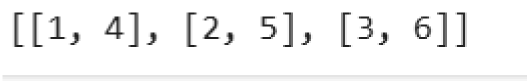
   ```
   matrix = [
    [1, 2, 3],
    [4, 5, 6]
   ]
   ```

   Output:

   ```
   [
   [1, 4],
   [2, 5],
   [3, 6]
   ]
   ```

```
matrix = [
    [1, 2, 3],
    [4, 5, 6]
]


transposed_matrix = [[matrix[j][i] for j in range(len(matrix))] for i in range(len(matrix[0]))]

print(transposed_matrix)
```

OUTPUT:



```
[[1, 4], [2, 5], [3, 6]]
```

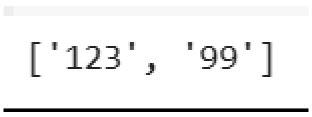
PROGRAM 14:

Given a list of strings, extract digits from each string and store them in a new list.

Input: ["abc123", "hello99"] → Output: ["123", "99"]

```
input_list = ["abc123", "hello99"]
output_list = [''.join(filter(str.isdigit, string)) for string in input_list]
print(output_list)
```

OUTPUT:



```
['123', '99']
```

DEPARTMENT OF CSE		
Program	10	
Output	5	
Viva-Voce	5	
Total	20	