SUPERIOR UNIVERSITY

Data Structure and Algorithm (Lab)

Assignment - 1

Name:

Ali Maqsood.

Roll no:

SU92-BSAIM-F23-050.

Department:

Software Engineering Department.

Program:

Artificial Intelligence.

Section:

BSAI-3A

Task # 1:

Implement a stack using a class in Python. The stack should support the following operations:

- push(element): Adds an element to the top of the stack.
- pop(): Removes and returns the top element from the stack.
- peek(): Returns the top element without removing it.
- is_empty(): Returns True if the stack is empty, False otherwise.
- size(): Returns the number of elements in the stack.

Code:

```
class stack():
    def __init__(self):
        self.stack=[]
    def push(self):
        value=input("Enter the element you want to Push: ")
        self.stack.append(value)
    def pop(self):
        print("Popped element is: ",self.stack.pop())
        print(f"The Top element is: {self.stack[-1]}")
    def peek(self):
        print(f"The Top element is: {self.stack[-1]}")
    def is_empty(self):
        if not self.stack:
            print("True.")
        else:
            print("False.")
    def size(self):
        print(f"The size of stack is: {len(self.stack)}.")
```

```
obj1=stack()
obj1.push()
obj1.push()
obj1.push()
obj1.pop()
obj1.peek()
obj1.size()
obj1.is_empty()
```

Output:

```
E:\Uni\3rd Semester\4) Data Structures & Algorithms (Lab)\Assignments\Assignment 1(Final)>python task.py
Enter the element you want to Push: 10
Enter the element you want to Push: 20
Enter the element you want to Push: 30
Popped element is:  30
The Top element is: 20
The Top element is: 20
The size of stack is: 2.
False.
```

Task # 2:

Implement a queue using a class in Python. The queue should support the following operations:

- enqueue(element): Adds an element to the end of the queue.
- dequeue(): Removes and returns the front element from the queue.
- front(): Returns the front element without removing it.
- is_empty(): Returns True if the queue is empty, False otherwise.
- size(): Returns the number of elements in the queue.

Code:

```
class queue():
    def __init__(self):
        self.queue=[]
    def enqueue(self):
        value=input("Enter the element you want to Enqueue: ")
        self.queue.append(value)
    def dequeue(self):
        print("Dequeued element is: ",self.queue.pop(0))
        print(f"The Front element is: {self.queue[0]}")
    def front(self):
        print(f"The Front element is: {self.queue[0]}")
    def is_empty(self):
        if not self.queue:
            print("True.")
        else:
            print("False.")
    def size(self):
        print(f"The size of queue is: {len(self.queue)}.")
```

```
obj2=queue()
obj2.enqueue()
obj2.enqueue()
obj2.enqueue()
obj2.dequeue()
obj2.front()
obj2.size()
obj2.is_empty()
```

Output:

```
E:\Uni\3rd Semester\4) Data Structures & Algorithms (Lab)\Assignments\Assignment 1(Final)>python task.py
Enter the element you want to Enqueue: 10
Enter the element you want to Enqueue: 20
Enter the element you want to Enqueue: 30
Dequeued element is:  10
The Front element is: 20
The Front element is: 20
The size of queue is: 2.
False.
```

Task # 3:

Objective: Implement the following sorting algorithms and compare their performance:

1. **Bubble Sort**
2. **Selection Sort**
3. **Insertion Sort**

For each sorting algorithm, write a function that takes a list of integers as input and returns a sorted list. Implement a performance comparison by sorting a list of 1000 random integers and measuring the execution time for each algorithm.

Code:

```
import random

import time

def bubble(list1):

    # print(f"Before Sort: {list1}.")

    for i in range(len(list1)-1):

        for j in range(len(list1)-1-i):

            if list1[j]>list1[j+1]:

                list1[j],list1[j+1]=list1[j+1],list1[j]

    # print(f"After Sort: {list1}.")

# bubble([64, 34, 25, 12, 22, 11, 90])


def selection(list2):

    # print(f"Before Sort: {list2}.")

    for i in range(len(list2)-1):

        small=i

        for j in range(i+1,len(list2)):

            if list2[j]<list2[small]:

                small=j
```

```
        list2[i],list2[small]=list2[small],list2[i]
    # print(f"After Sort: {list2}.")
# selection([64, 34, 25, 12, 22, 11, 90])

def insertion(list3):
    # print(f"Before Sort: {list3}.")
    for i in range(1,len(list3)):
        key=list3[i]
        j=i-1
        while j>=0 and key<list3[j]:
            list3[j+1]=list3[j]
            j=j-1
        list3[j+1]=key
    # print(f"After Sort: {list3}.")
# insertion([64, 34, 25, 12, 22, 11, 90])

def comparison():
    final_arr=[]
    for i in range(1000):
        final_arr.append(random.randint(1,1000))

    bubble_copy=final_arr.copy()
    start=time.time()
    bubble(bubble_copy)
    end=time.time()
    print(f"Bubble Sort Time: {end-start: .3f} seconds.")
```

```
    selection_copy=final_arr.copy()
    start=time.time()
    selection(selection_copy)
    end=time.time()
    print(f"Selection Sort Time: {end-start: .3f} seconds.")

    insertion_copy=final_arr.copy()
    start=time.time()
    insertion(insertion_copy)
    end=time.time()
    print(f"Insertion Sort Time: {end-start: .3f} seconds.")
comparison()
```

Output:

```
E:\Uni\3rd Semester\4) Data Structures & Algorithms (Lab)\Assignments\Assignment 1(Final)>python task.py
Bubble Sort Time:  0.080 seconds.
Selection Sort Time:  0.040 seconds.
Insertion Sort Time:  0.043 seconds.
```

Task # 4:

Write a function that inserts an element into a sorted list while maintaining the sorted order. The function should return the updated sorted list.

Code:

```
def sorted_insertion(arr):
    element=int(input("Enter the element to be inserted: "))
    arr.append(element)
    for i in range(len(arr) -1, 0,-1):
        if arr[i]<arr[i-1]:
            arr[i], arr[i-1]=arr[i-1], arr[i]
        else:
            break
    print(arr)
sorted_list = [10,20,30,40,50]
sorted_insertion(sorted_list)
```

Output:

```
E:\Uni\3rd Semester\4) Data Structures & Algorithms (Lab)\Assignments\Assignment 1(Final)>python task.py
Enter the element to be inserted: 35
[10, 20, 30, 35, 40, 50]
```