```
Print(“****************Stack***************“)

Class CustomStack:
   Elements = []

   Def __init__(self):
      Self.container = []

   Def push(self, item):
      Self.container = [item] + self.container

   Def pop(self):
      Return self.container.pop(0)

Stack_instance = CustomStack()
Stack_instance.push(20)
Stack_instance.push(30)
Stack_instance.push(40)
Stack_instance.push(50)
Print(stack_instance.container)
Stack_instance.pop()
Print(stack_instance.container)

Print(“********************Queue*****************“)

Class CustomQueue:
   Elements = []

   Def __init__(self):
```

```
        Self.container = []

    Def enqueue(self, item):
        Self.container = self.container + [item]

    Def dequeue(self):
        Return self.container.pop(0)

Queue_instance = CustomQueue()
Queue_instance.enqueue(20)
Queue_instance.enqueue(30)
Queue_instance.enqueue(40)
Queue_instance.enqueue(50)
Print(queue_instance.container)
Queue_instance.dequeue()
Print(queue_instance.container)

Print("**********Binary Search Algorithm**********")

Data = [6, 12, 17, 23, 38, 45, 77, 84, 90]
Search_target = 45
Low = 0
High = len(data) – 1
Found = False

While low <= high:
    Middle = int((low + high) / 2)
    If data[middle] == search_target:
        Found = True
```

```
        Break
    Elif data[middle] < search_target:
        Low = middle + 1
    Else:
        High = middle – 1

If found:
    Print(“Target found Successfully!”)
Else:
    Print(“Target not found!”)
```