Question:1
Implement Stack using Python.

python.py > ...

```python
class Stack:
    def __init__(self):
        self.stack = []

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

    def pop(self):
        if not self.is_empty():
            return self.stack.pop()
        return None

    def peek(self):
        if not self.is_empty():
            return self.stack[-1]
        return None

    def is_empty(self):
        return len(self.stack) == 0

    def size(self):
        return len(self.stack)

s = Stack()
print("Is stack empty?", s.is_empty())
s.push(10)
s.push(20)
s.push(30)
print("Stack size is:", s.size())
print("Stack top is:", s.peek())
print("Popped:", s.pop())
print("Stack size is:", s.size())
print("Is stack empty?", s.is_empty())
```

Question:2

Implement Queue using Python.

python.py > ...

```python
from collections import deque

class Queue:
    def __init__(self):
        self.queue = deque()

    def enqueue(self, item):
        self.queue.append(item)

    def dequeue(self):
        if not self.is_empty():
            return self.queue.popleft()
        return None

    def front(self):
        if not self.is_empty():
            return self.queue[0]
        return None

    def is_empty(self):
        return len(self.queue) == 0

    def size(self):
        return len(self.queue)


q = Queue()
q.enqueue(10)
q.enqueue(20)
q.enqueue(30)
print("Queue front is:", q.front())
print("Dequeued:", q.dequeue())
print("Queue size is:", q.size())
```

Question:3
Binary Search in python.

python.py > ...

```python
def binary_search(sorted_list, target):
    start_index = 0
    end_index = len(sorted_list) - 1

    while start_index <= end_index:
        middle_index = (start_index + end_index) // 2
        middle_value = sorted_list[middle_index]

        if middle_value == target:
            return middle_index
        elif middle_value < target:
            start_index = middle_index + 1
        else:
            end_index = middle_index - 1

    return -1

if __name__ == "__main__":
    numbers = [1, 3, 4, 6, 8, 10, 13, 15, 18, 20]
    target_value = 14
    result_index = binary_search(numbers, target_value)
    if result_index != -1:
        print(f"Target {target_value} found at index {result_index}.")
    else:
        print(f"Target {target_value} not found in the list.")
```