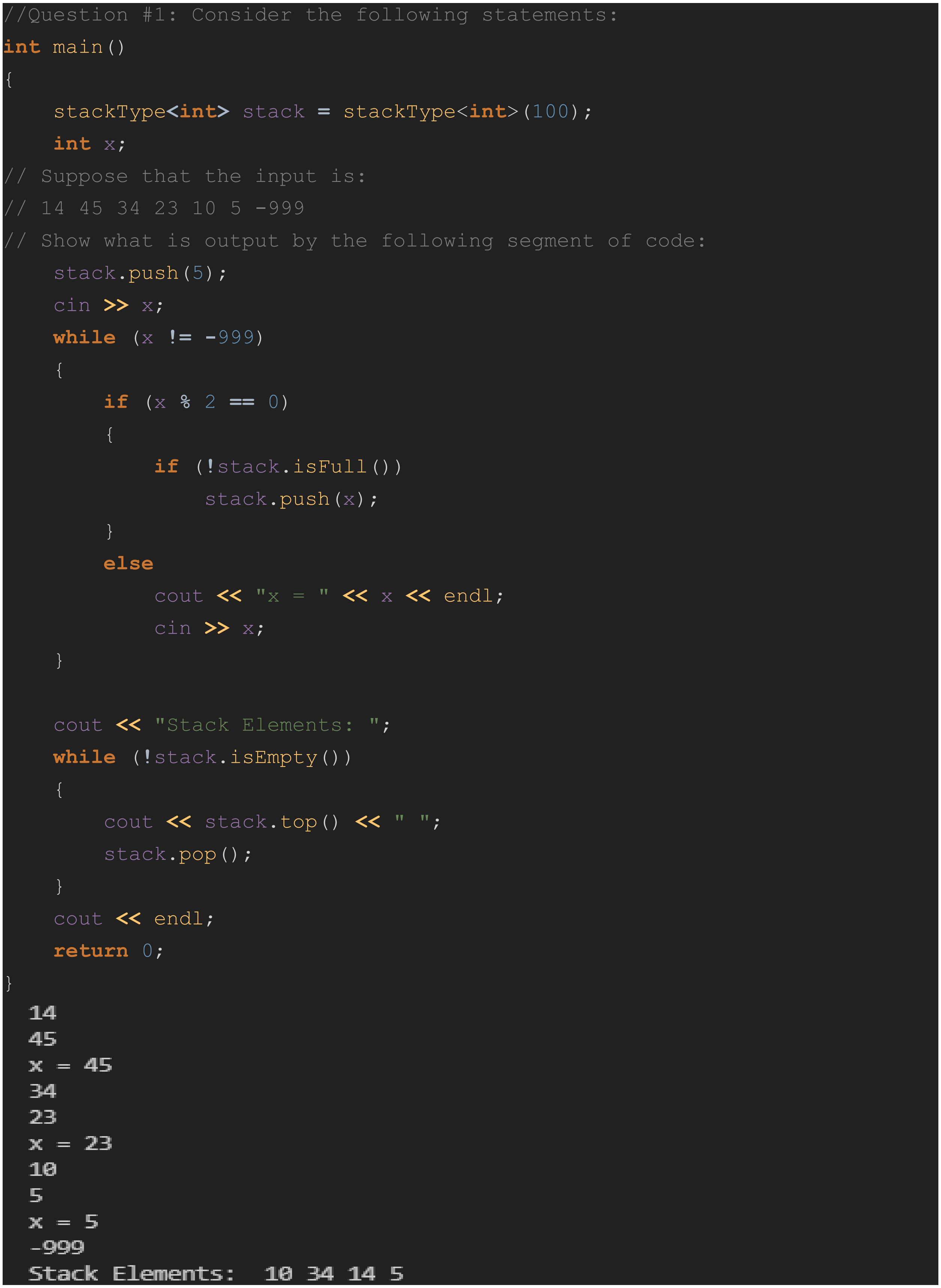
QUESTION 1:

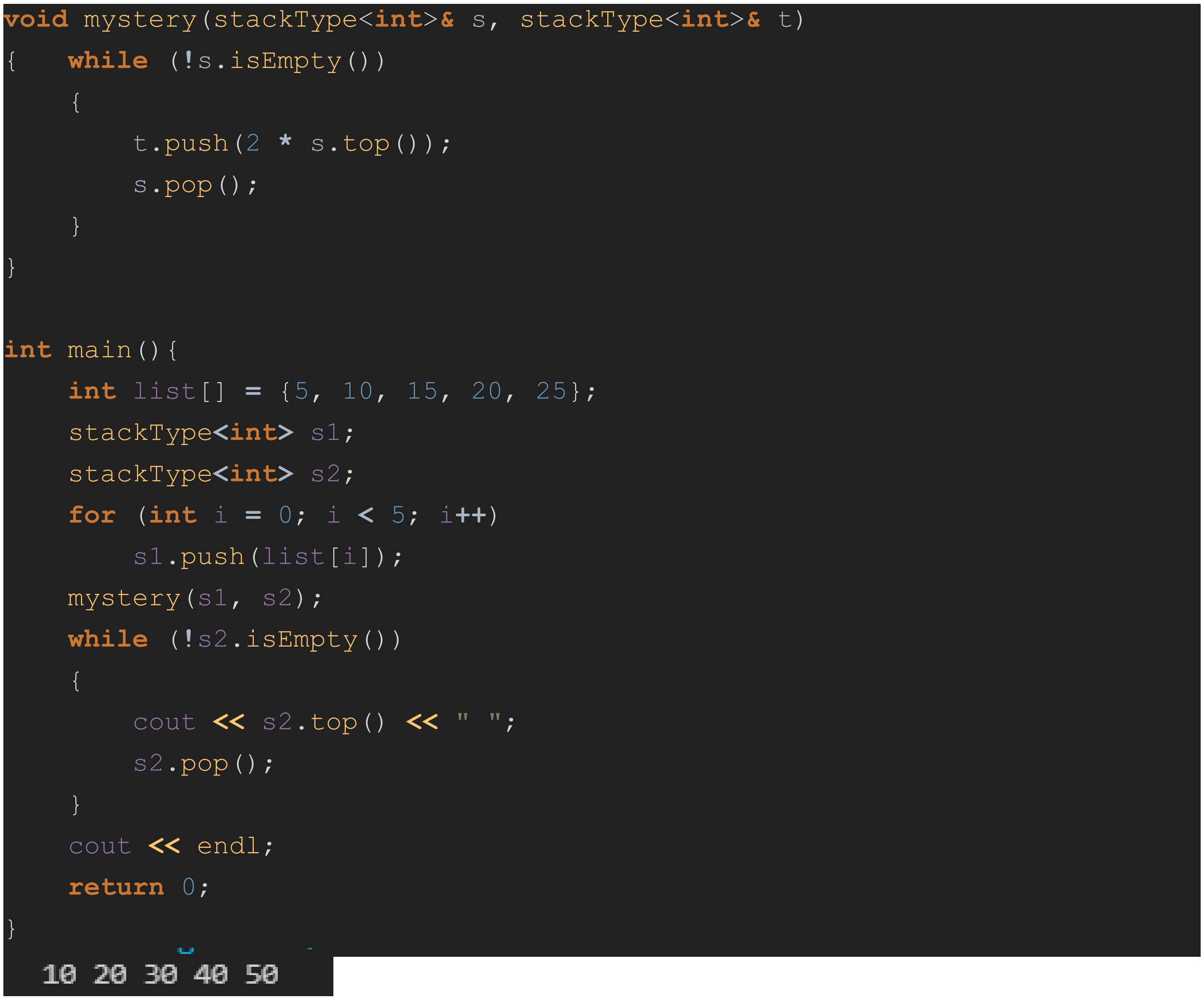


```
//Question #1: Consider the following statements:
int main()
{
    stackType<int> stack = stackType<int>(100);
    int x;
// Suppose that the input is:
// 14 45 34 23 10 5 -999
// Show what is output by the following segment of code:
    stack.push(5);
    cin >> x;
    while (x != -999)
    {
        if (x % 2 == 0)
        {
            if (!stack.isFull())
                stack.push(x);
        }
        else
            cout << "x = " << x << endl;
            cin >> x;
    }

    cout << "Stack Elements: ";
    while (!stack.isEmpty())
    {
        cout << stack.top() << " ";
        stack.pop();
    }
    cout << endl;
    return 0;
}
```

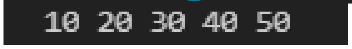
```
  14
  45
  x = 45
  34
  23
  x = 23
  10
  5
  x = 5
  -999
  Stack Elements:  10 34 14 5
```

QUESTION 2



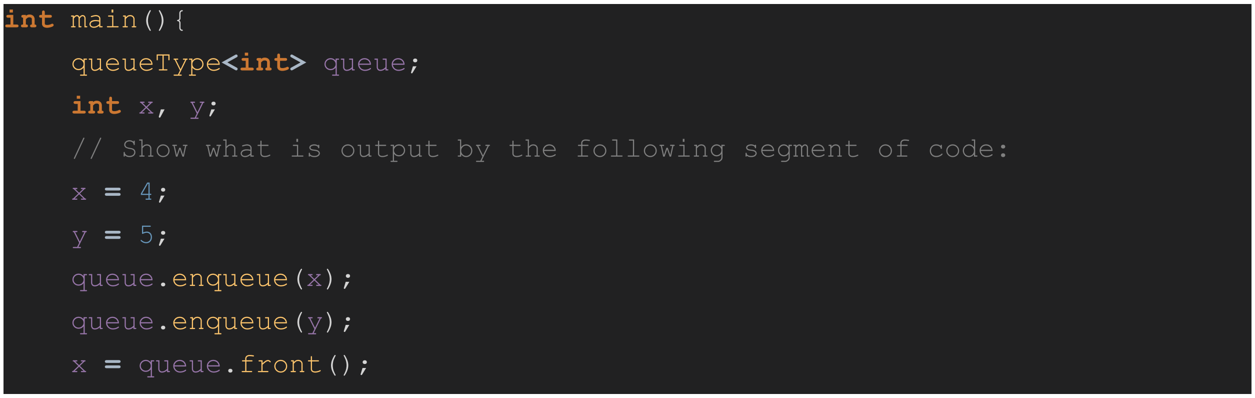
```
void mystery(stackType<int>& s, stackType<int>& t)
{    while (!s.isEmpty())
     {
         t.push(2 * s.top());
         s.pop();
     }
}

int main(){
    int list[] = {5, 10, 15, 20, 25};
    stackType<int> s1;
    stackType<int> s2;
    for (int i = 0; i < 5; i++)
        s1.push(list[i]);
    mystery(s1, s2);
    while (!s2.isEmpty())
    {
        cout << s2.top() << " ";
        s2.pop();
    }
    cout << endl;
    return 0;
}
```

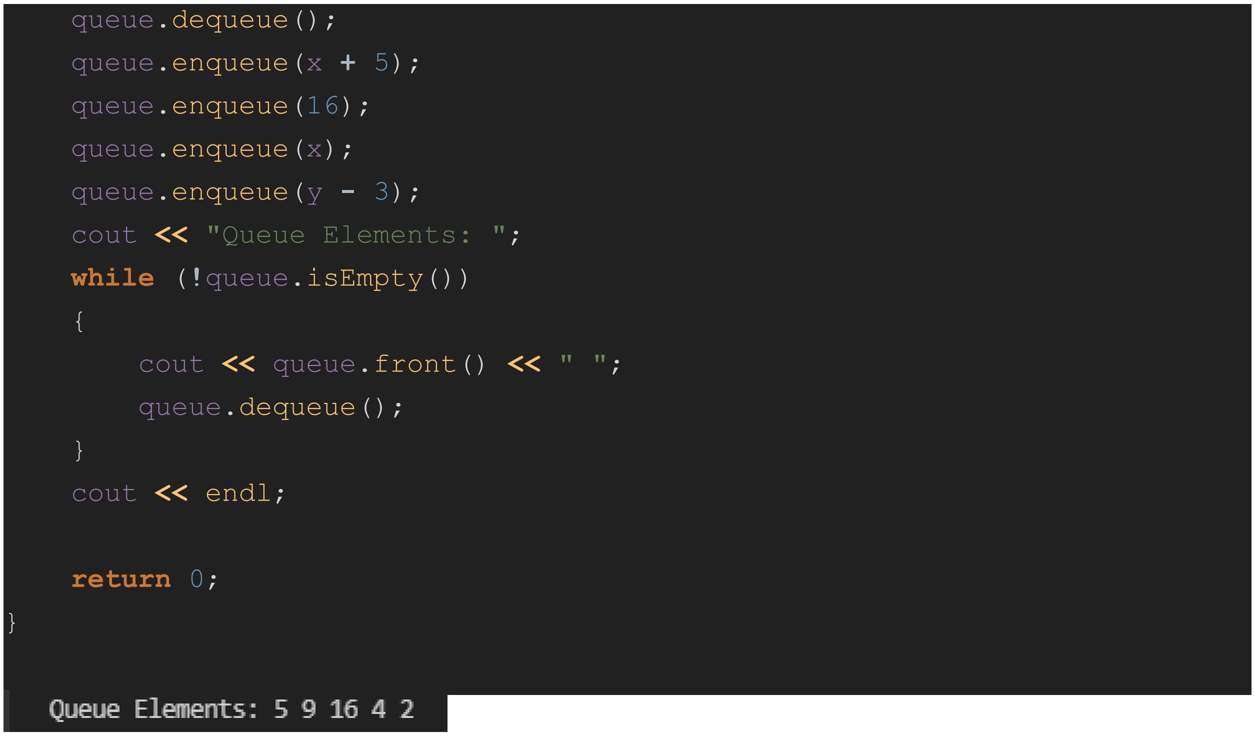


```
10 20 30 40 50
```

QUESTION 3:

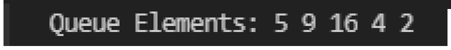


```
int main(){
    queueType<int> queue;
    int x, y;
    // Show what is output by the following segment of code:
    x = 4;
    y = 5;
    queue.enqueue(x);
    queue.enqueue(y);
    x = queue.front();
```



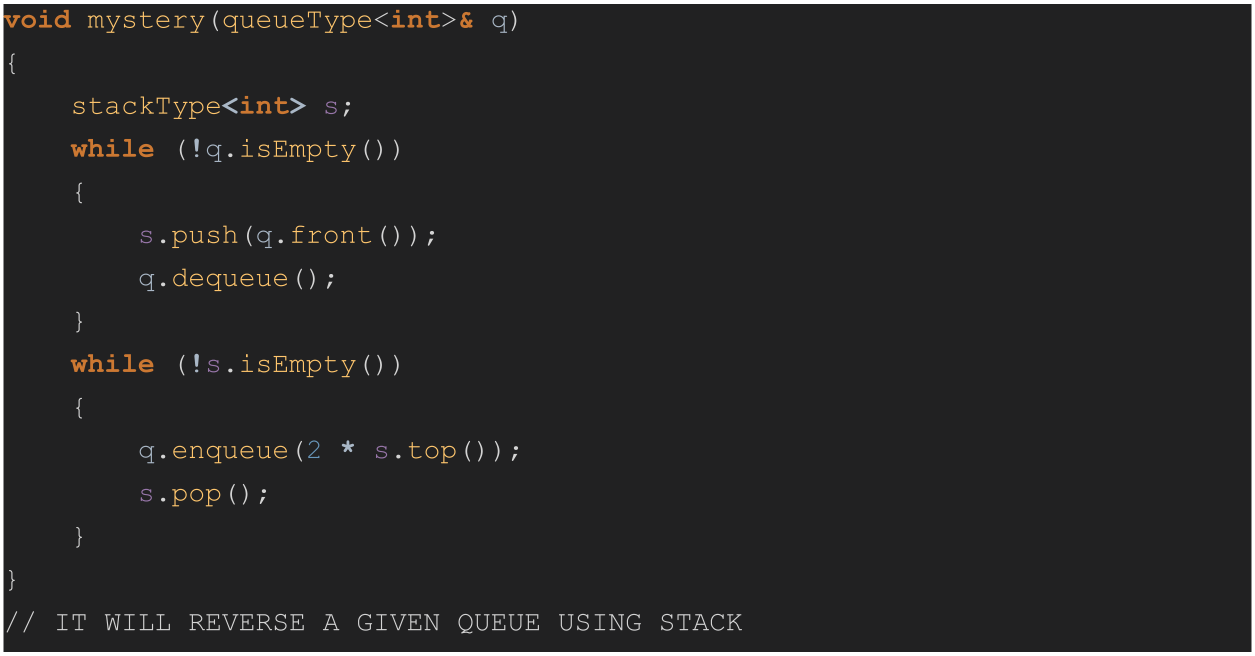
```
    queue.dequeue();
    queue.enqueue(x + 5);
    queue.enqueue(16);
    queue.enqueue(x);
    queue.enqueue(y - 3);
    cout << "Queue Elements: ";
    while (!queue.isEmpty())
    {
        cout << queue.front() << " ";
        queue.dequeue();
    }
    cout << endl;

    return 0;
}
```



```
Queue Elements: 5 9 16 4 2
```

QUESTION 4: What does the following function do?



```
void mystery(queueType<int>& q)
{
    stackType<int> s;
    while (!q.isEmpty())
    {
        s.push(q.front());
        q.dequeue();
    }
    while (!s.isEmpty())
    {
        q.enqueue(2 * s.top());
        s.pop();
    }
}
// IT WILL REVERSE A GIVEN QUEUE USING STACK
```

QUESTION #8:

Suppose that queue is a queueType object and the size of the array implementing queue is 100. Also, suppose that the value of queueFront is 50 and the value of queueRear is 99.

a. What are the values of queueFront and queueRear after adding an element to queue?

ANSWER: Suppose the input value is 70.
queueFront will be the same. It will remain 50.
queueRear will change. It will be updated to 70 from 99.

b. What are the values of queueFront and queueRear after removing an element from queue?

ANSWER: queueFront will change. It will change to the second item in the queue.
queueRear will remain the same. It will be 99.

QUESTION #9:

Suppose that queue is a queueType object and the size of the array implementing queue is 100. Also, suppose that the value of queueFront is 99 and the value of queueRear is 25.

a. What are the values of queueFront and queueRear after adding an element to queue?

ANSWER: Suppose the input value is 70.
queueFront will be the same. It will remain 99.
queueRear will change. It will be updated to 70 from 25.

b. What are the values of queueFront and queueRear after removing an element from queue?

ANSWER: queueFront will change. It will change to the second item in the queue.
queueRear will remain the same. It will be 25.