Chandigarh College of Engineering & Technology (Degree Wing)
Department of Computer Science & Engineering

DATA STRUCTURES REPORT

Abdul Rahim (CO20301)

Files included:

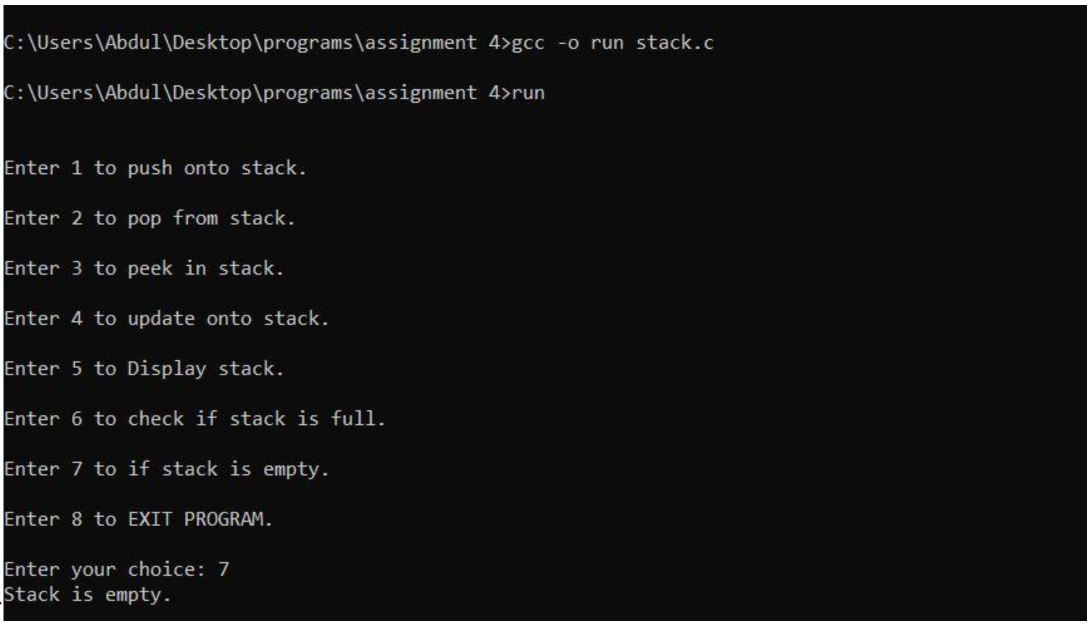
Stack.c , report

Data structure used:

Stack implementation of array.

4.Outputs: Following are the screenshots representing the outputs of different sets of data used in each method. The program was executed on Command prompt.

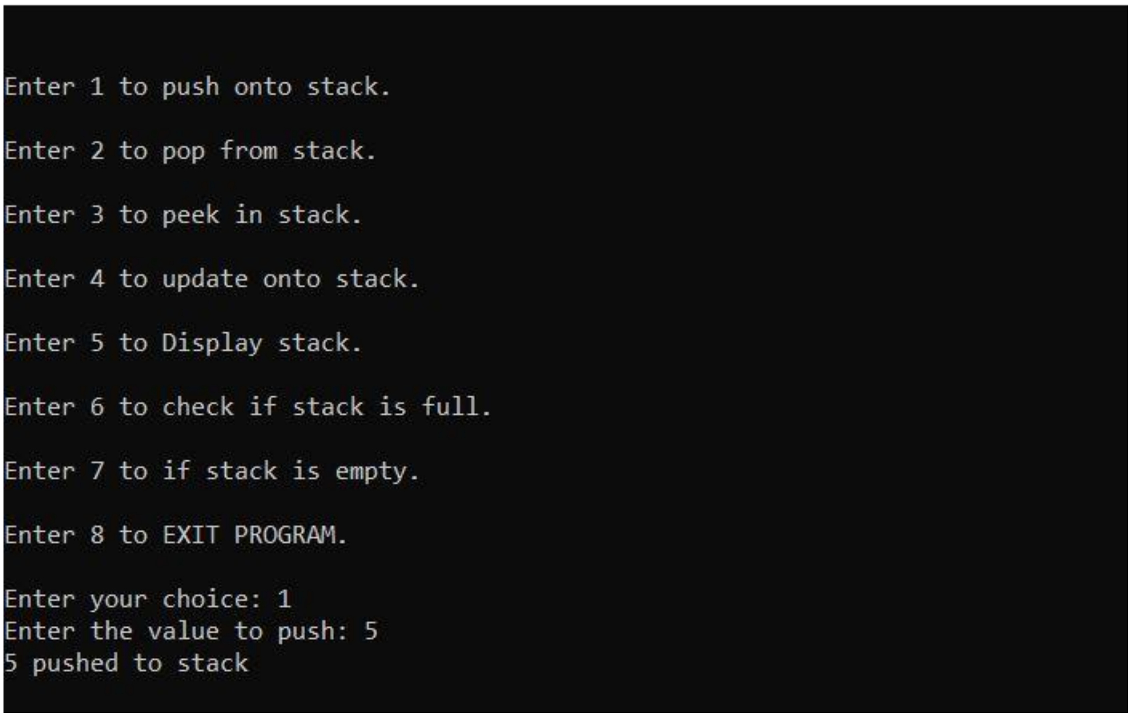
Is EMPTY method():



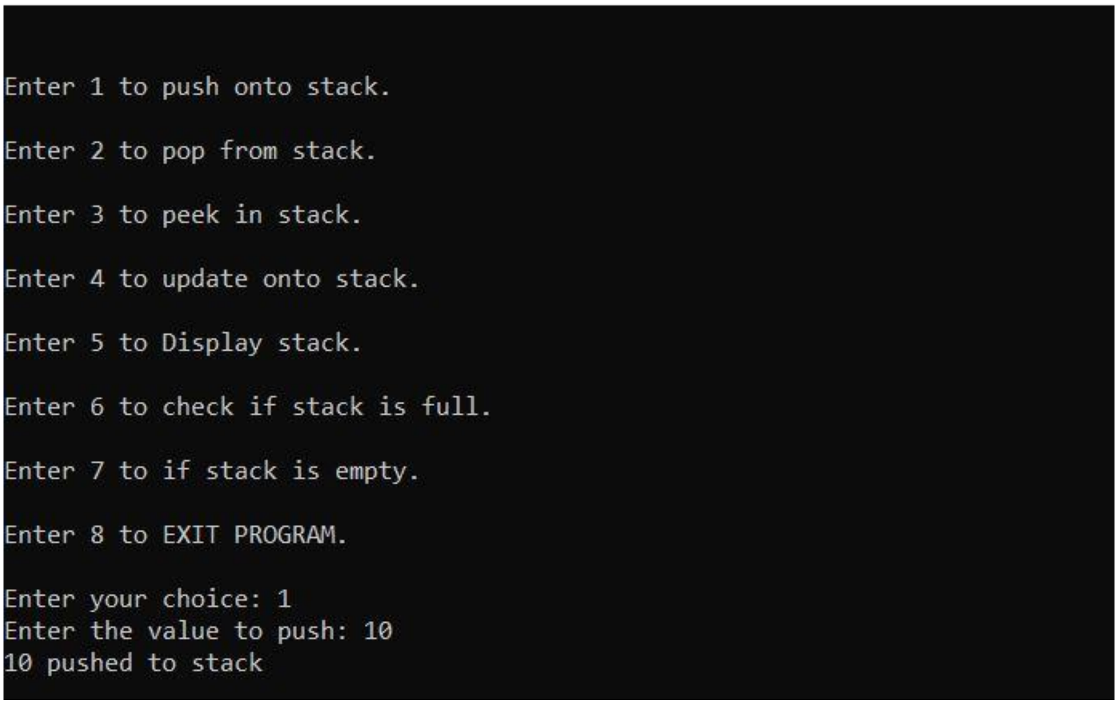
```
C:\Users\Abdul\Desktop\programs\assignment 4>gcc -o run stack.c

C:\Users\Abdul\Desktop\programs\assignment 4>run


Enter 1 to push onto stack.

Enter 2 to pop from stack.

Enter 3 to peek in stack.

Enter 4 to update onto stack.

Enter 5 to Display stack.

Enter 6 to check if stack is full.

Enter 7 to if stack is empty.

Enter 8 to EXIT PROGRAM.

Enter your choice: 7
Stack is empty.
```

`

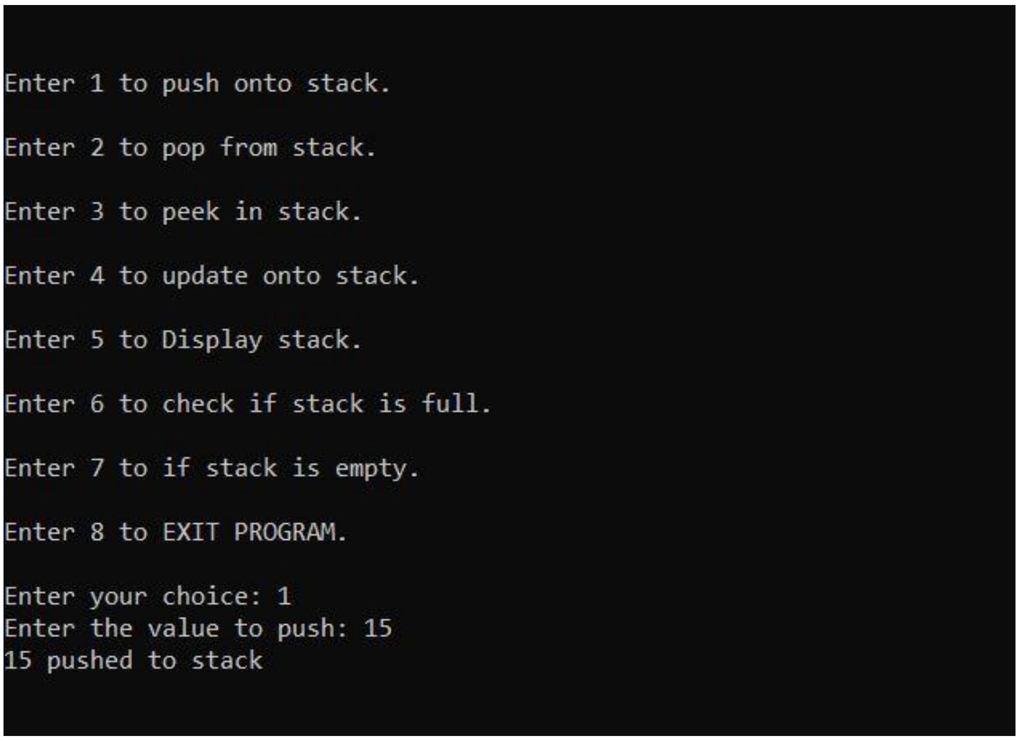
Push method:



```


Enter 1 to push onto stack.

Enter 2 to pop from stack.

Enter 3 to peek in stack.

Enter 4 to update onto stack.

Enter 5 to Display stack.

Enter 6 to check if stack is full.

Enter 7 to if stack is empty.

Enter 8 to EXIT PROGRAM.

Enter your choice: 1
Enter the value to push: 5
5 pushed to stack
```

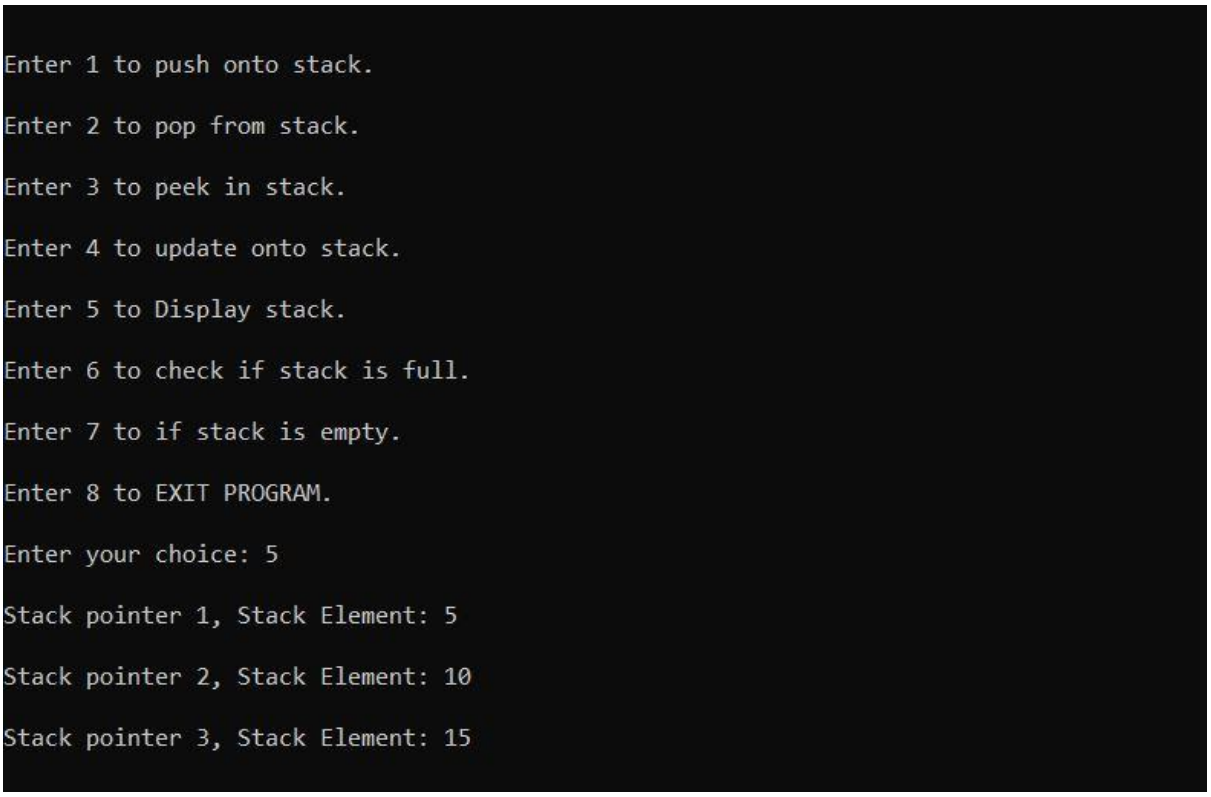


```
Enter 1 to push onto stack.

Enter 2 to pop from stack.

Enter 3 to peek in stack.

Enter 4 to update onto stack.

Enter 5 to Display stack.

Enter 6 to check if stack is full.

Enter 7 to if stack is empty.

Enter 8 to EXIT PROGRAM.

Enter your choice: 1
Enter the value to push: 10
10 pushed to stack
```



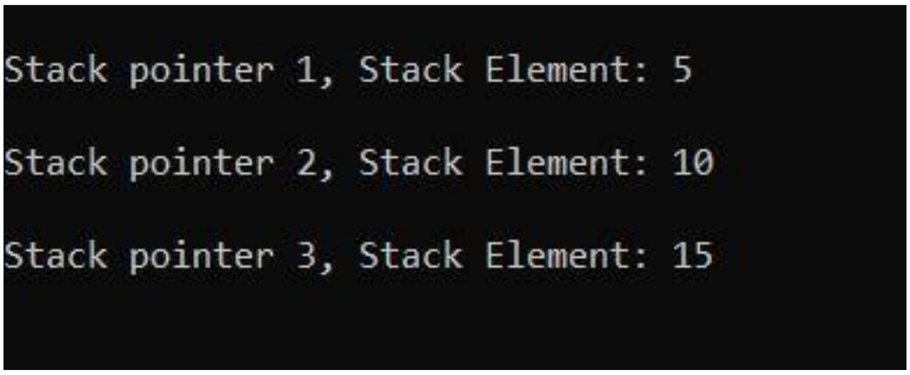
```
Enter 1 to push onto stack.

Enter 2 to pop from stack.

Enter 3 to peek in stack.

Enter 4 to update onto stack.

Enter 5 to Display stack.

Enter 6 to check if stack is full.

Enter 7 to if stack is empty.

Enter 8 to EXIT PROGRAM.

Enter your choice: 1
Enter the value to push: 15
15 pushed to stack
```

Display:



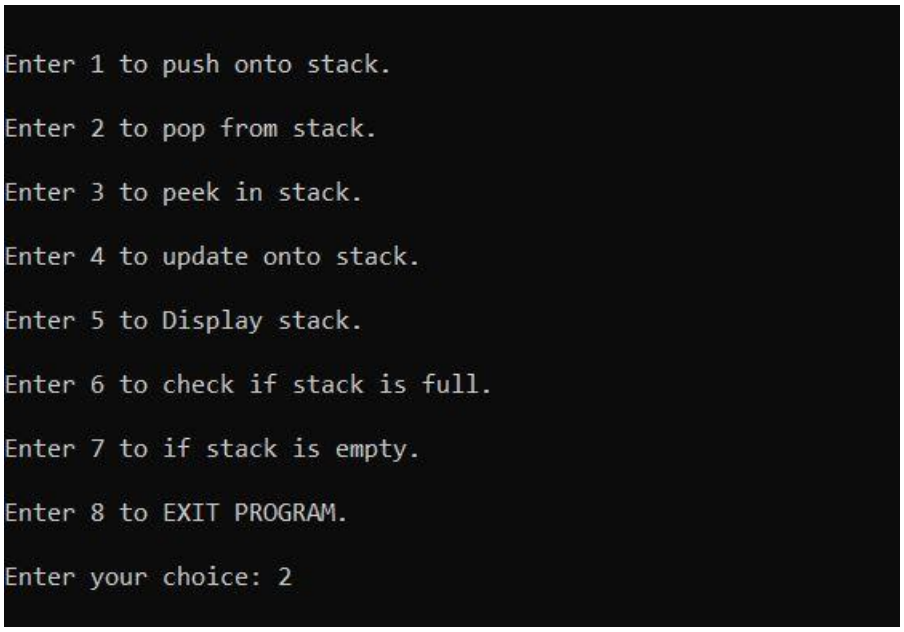
```
Enter 1 to push onto stack.

Enter 2 to pop from stack.

Enter 3 to peek in stack.

Enter 4 to update onto stack.

Enter 5 to Display stack.

Enter 6 to check if stack is full.

Enter 7 to if stack is empty.

Enter 8 to EXIT PROGRAM.

Enter your choice: 5

Stack pointer 1, Stack Element: 5

Stack pointer 2, Stack Element: 10

Stack pointer 3, Stack Element: 15
```

Stack created:

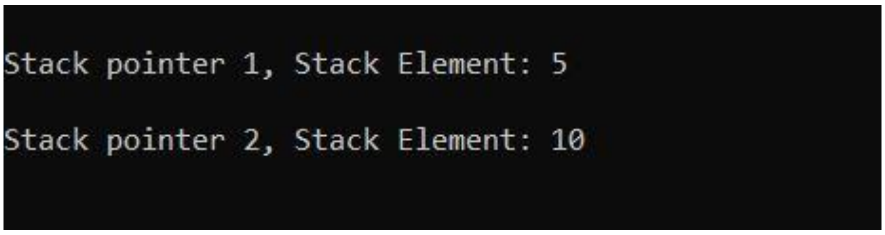


```
Stack pointer 1, Stack Element: 5

Stack pointer 2, Stack Element: 10

Stack pointer 3, Stack Element: 15
```

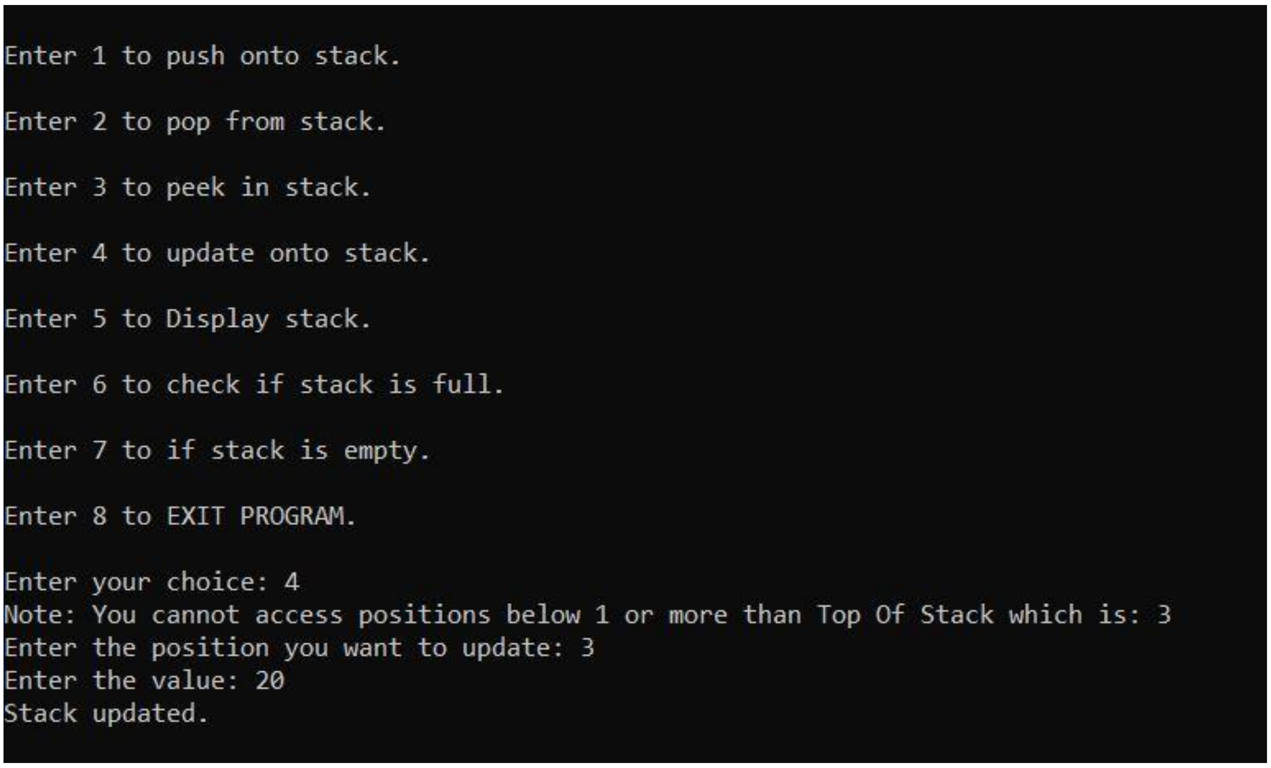
Pop method:


```
Enter 1 to push onto stack.

Enter 2 to pop from stack.

Enter 3 to peek in stack.

Enter 4 to update onto stack.

Enter 5 to Display stack.

Enter 6 to check if stack is full.

Enter 7 to if stack is empty.

Enter 8 to EXIT PROGRAM.

Enter your choice: 2
```

Displaying stack give**:**

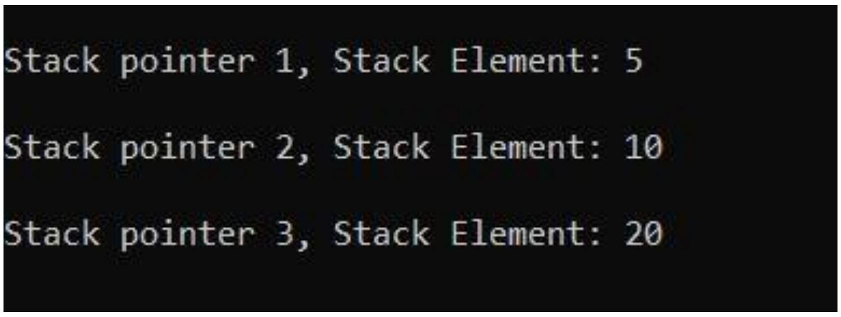

```
Stack pointer 1, Stack Element: 5

Stack pointer 2, Stack Element: 10
```

Update:



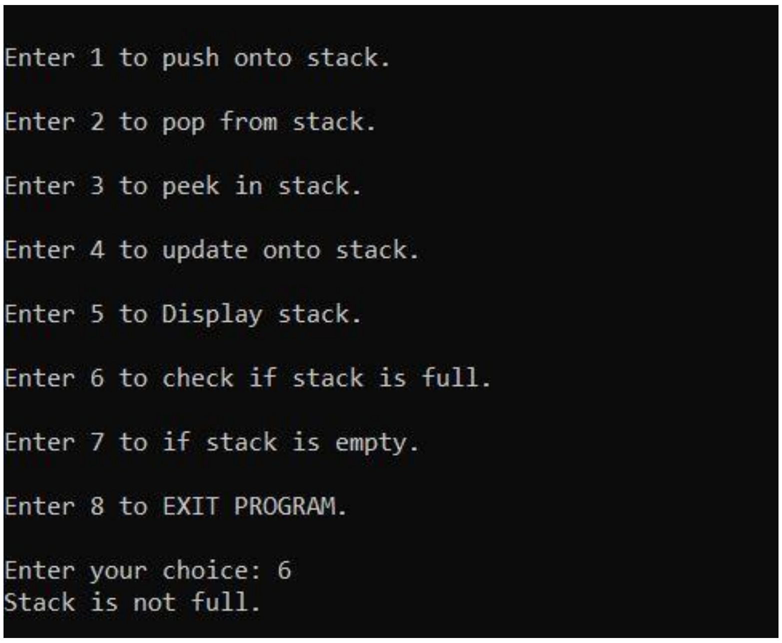
```
Enter 1 to push onto stack.

Enter 2 to pop from stack.

Enter 3 to peek in stack.

Enter 4 to update onto stack.

Enter 5 to Display stack.

Enter 6 to check if stack is full.

Enter 7 to if stack is empty.

Enter 8 to EXIT PROGRAM.

Enter your choice: 4
Note: You cannot access positions below 1 or more than Top Of Stack which is: 3
Enter the position you want to update: 3
Enter the value: 20
Stack updated.
```

Displaying updated stack:



```
Stack pointer 1, Stack Element: 5

Stack pointer 2, Stack Element: 10

Stack pointer 3, Stack Element: 20
```

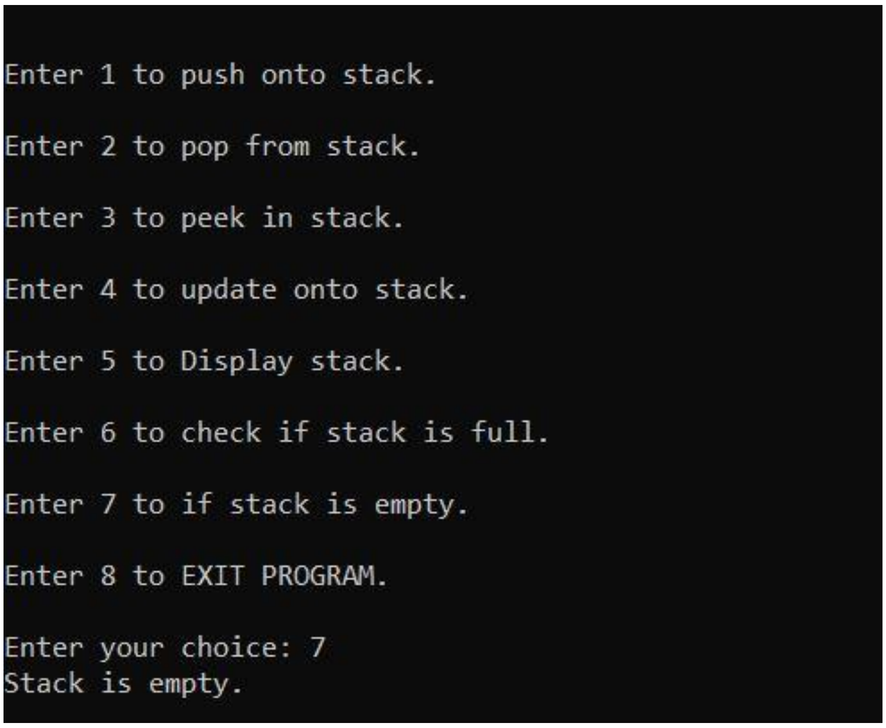
Is full method:



```
Enter 1 to push onto stack.

Enter 2 to pop from stack.

Enter 3 to peek in stack.

Enter 4 to update onto stack.

Enter 5 to Display stack.

Enter 6 to check if stack is full.

Enter 7 to if stack is empty.

Enter 8 to EXIT PROGRAM.

Enter your choice: 6
Stack is not full.
```

Is empty method();

After two pops stack was empty:



```
Enter 1 to push onto stack.

Enter 2 to pop from stack.

Enter 3 to peek in stack.

Enter 4 to update onto stack.

Enter 5 to Display stack.

Enter 6 to check if stack is full.

Enter 7 to if stack is empty.

Enter 8 to EXIT PROGRAM.

Enter your choice: 7
Stack is empty.
```

Conclusion:

A. Push Method: This method inserts an element to the top of stack. Since stack is based on LIFO (Last in First Out) operation, both push and pop operations take place at the top of stack only. Its time complexity is O(1) and so is the space complexity.

B. Pop Method: This method removes an element from the top of stack. Since stack is based on LIFO (Last in First Out) operation, both push and pop operations take place at the top of stack only. Its time complexity is O(1) and so is the space complexity.

C. Search (peep): It returns the value present at a particular index / position. It first pops out all the elements from the original stack and keep pushing the popped elements into a temporary stack till it reaches at desired index. After that it returns the value present at that index. Then, the original stack is reconstructed from the temporary stack by popping each element from the temporary stack and pushing it into the original stack. Time Complexity is O(n) and space complexity is also S(n) since we need to create a temporary stack of same size as original stack.

D. Update: It works in the same way as peep but instead of returning the value at index, it assigns it with a new desired value. It first pops out all the elements from the original stack and keep pushing the popped elements into a temporary stack till it reaches at desired index. After that assigns the value at that index with a new desired value. Then, the original stack is reconstructed from the temporary stack by popping each element from the temporary stack and pushing it into the original stack. Time Complexity is O(n) and space complexity is also S(n) since we need to create a temporary stack of same size as original stack.

E. isEmpty: This method returns true if the stack is empty and false if its not empty. It checks so by checking the value of tos (top of stack). If tos == -1, it returns true. Else, returns false.

E. isFull: This method returns true if the stack is full and false if it is not full. It checks so by checking the value of tos (top of stack). If tos == size (maximum size of the stack), it returns true. Else, returns false.