Avila, Sean Rendel S.

Basic Stack Operations

```
#include <iostream>
class Stack {
  private:
    int top;
    int maxSize;
    int* stackArray;

    public:
      Stack(int size) {
        maxSize = size;
        stackArray = new int[maxSize];
        top = -1;
      }

      ~Stack() {
        delete[] stackArray;
      }
      bool isEmpty() {
        return top == -1;
      }
      bool isFull() {
        return top == maxSize -1;
      }
      void push(int value) {
        if (!isFull()){
          top++;
          stackArray[top] = value;
          std::cout << "Pushed " << value << " onto the stack.\n";
        } else {
            std::cout << "Stack is full.\n";
        }
    }
    void pop() {
        if (!isEmpty()) {
            int poppedValue = stackArray[top];
            top--;
            std::cout << "Popped " << poppedValue << " from the stack.\n";
        } else {
            std::cout << "Stack is empty.\n";
        }
    }

    int peek() {
    if (!isEmpty()) {
            return stackArray[top];
        } else {
            std::cout << "Stack is empty.\n";
            return -1;
        }
    }

};
int main() {
  Stack stack(5);

  std::cout << "Is empty?: " << stack.isEmpty() << std::endl;
  stack.push(10);
  stack.push(20);
  stack.push(30);
  std::cout << "Is full?: " << stack.isFull() << std::endl;
  stack.push(40);
  stack.push(50);
  std::cout << "Top element: " << stack.peek() << std::endl;
  std::cout << "Is full?: " << stack.isFull() << std::endl;
  stack.pop();
  stack.pop();
  std::cout << "Top element after popping: " << stack.peek() << std::endl;
  stack.push(60);
```

```
    return 0;
}
```