Overview of Implemented Functions

StackLinked Functions

```
StackLinked(int maxNumber = Stack<DataType>::MAX_STACK_SIZE);
```

- Didn’t use the maxNumber parameter for anything, just set the top pointer to nullptr

```
void deepCopy(StackNode * curr);
```

- Helper function used in the copy constructor and assignment operator. Takes the top pointer of another stack, uses recursion to go to the bottom of the stack, and then pushes items from that stack in reverse order.

```
StackLinked(const StackLinked& other);
```

- Calls deepCopy with the top pointer of the “other” stack

```
StackLinked& operator=(const StackLinked& other);
```

- Checks for self assignment, clears the current stack if it isn’t empty already, then calls deepCopy

```
~StackLinked();
```

- Calls clear()

```
void push(const DataType& newDataItem) throw (logic_error);
```

- Creates new StackNode, sets it to top, and assigns its “next” pointer to the old top

```
DataType pop() throw (logic_error);
```

- Throws error if the stack is empty. If not, saves the top item, deletes the old top, sets the new top, and then returns the saved item.

```
void clear();
```

- Calls pop() until the top pointer is null

```
bool isEmpty() const;
```

- Checks if top is equal to nullptr

```
bool isFull() const;
```

- Returns false

```
void showStructure() const;
```

- Copied from show6.cpp

```
bool hasOne() const;
```

- Used in the postfix program, checks if there is only 1 item in the stack

```
StackNode(const DataType& nodeData, StackNode* nextPtr);
```

- Assigns the data and pointer members with the given input

StackArray Functions

```
StackArray(int maxNumber = Stack<DataType>::MAX_STACK_SIZE);
```

- Sets maxSize to maxNumber, top to -1, and allocates an array with size maxNumber
- **Does not** check if maxNumber is negative

```
StackArray(const StackArray& other);
```

- Sets maxSize the same as the other stack's, and allocates an array of the same size as the other stack, then copies all information

```
StackArray& operator=(const StackArray& other);
```

- Checks for self assignment, then performs the same operation as the copy constructor

```
~StackArray();
```

- Deletes the array

```
void push(const DataType& newDataItem) throw (logic_error);
```

- If the array is full, throws an error, if not, adds the item and updates the top of the stack

```
DataType pop() throw (logic_error);
```

- If empty, throws error, otherwise returns the item and updates the top

```
void clear();
```

- Sets the top to -1

```
bool isEmpty() const;
```

- Checks if top is equal to -1

```
bool isFull() const;
```

- Checks if top is equal to maxSize-1

```
void showStructure() const;
```

- Copied from show6.cpp

```
bool hasOne() const;
```

- Used in the postfix program, checks if there is only 1 item in the stack

Postfix Program

1. Uses getline() to put the input into a string
2. Converts the string into stringstream
3. Uses a loop to extract smaller strings from the stringstream
    a. If its a number, add it to the stack
    b. If its an operator, process the operation
4. If, at the end of the expression, there is only 1 item left in the stack, then the expression is complete

Delimiters Program

1. Goes through each character in the string
    a. If it’s an opening bracket, push it onto the stack
    b. If it’s a closing bracket, determine what type it is and see if it matches the current type as the top of the stack
        i. If it is, pop the top item off, otherwise the expression is not correctly delimited
2. Once all the characters are processed, there should be nothing else left on the stack