Activity No. 4	
STACKS	
Course Code: CPE010	**Program:** Computer Engineering
Course Title: Data Structures and Algorithms	**Date Performed: 10/04/2024**
Section: CPE21S4	**Date Submitted: 10/04/2024**
Name(s):Santos, Ma. Kassandra Nicole D.	**Instructor: Maam Sayo**

6. Output

OUTPUT	OBSERVATION/EXPLANATION

C/C++

```
#include <iostream>
#include <stack> // Calling Stack from the STL
using namespace std;

int main() {
    stack<int> newStack; // Create a stack of integers
    newStack.push(3); // Adds 3 to the stack
    newStack.push(8);
    newStack.push(15);

    // Check if the stack is empty
    cout << "Stack Empty? " << (newStack.empty() ? "Yes" : "No") << endl;

    // Display the size of the stack
    cout << "Stack Size: " << newStack.size() << endl;

    // Display the topmost element of the stack
    cout << "Top Element of the Stack: " << newStack.top() << endl;

    // Remove the topmost element of the stack
    newStack.pop();
    cout << "Top Element of the Stack after pop: " << newStack.top() << endl;
    cout << "Stack Size after pop: " << newStack.size() << endl;

    return 0;
}
```

The code is comprised of 5 objectives.

1. Check if the stack is empty or not
2. check the size of the array
3. Peek to see if what is at the top of the stack
4. Peek to see if what is at the top of the stack after popping or removing the prior element
5. check the size of the array after popping the prior element

C/C++

```
    stack<int> newStack; // Create a stack of integers
    newStack.push(3); // Adds 3 to the stack
    newStack.push(8);
    newStack.push(15);

// this part of the code adds the element to the stack known as pushing, in that order, if we were to display the output, it would be; 15,8,3

  cout << "Stack Empty? " << (newStack.empty() ? "Yes" : "No") << endl;

// this checks whether the stack is empty of (which in this case is not hence that output says No). empty() is a function to check if the stack is empty or not

    cout << "Stack Size: " << newStack.size() << endl;


    // Display the size of the stack. size () is a function return the number of elements currently in the stack
```

```
Stack Empty? No
Stack Size: 3
Top Element of the Stack: 15
Top Element of the Stack after pop: 8
Stack Size after pop: 2
```

```
cout << "Top Element of the Stack: " <<
newStack.top() << endl;

// top() function return to the topmost
element. since stacks follows first in
first out, the topmost element would be
15

newStack.pop();
    cout << "Top Element of the Stack
after pop: " << newStack.top() << endl;

//as mentioned, stacks follows first in
first out. Since we used the pop()
function to remove the current topmost
function, it will now display the new
topmost function which is 8

    cout << "Stack Size after pop: " <<
newStack.size() << endl;

// since we remove 15 from the stacks
and what remains is 8 and 3, the size of
the stacks will now be 2
```

TABLE 4.1 ILO A

OUTPUT	OBSERVATION/EXPLANATION

C/C++

```
#include <iostream>
using namespace std;

const size_t maxCap = 100;
int stack[maxCap];
int top = -1, i, newData;

void push();
void pop();
void Top();
bool isEmpty();
void displayStack();

int main() {
    int choice;
```

C/C++

```
// ADDED CODE

     case 5:
                displayStack();
                break;
            case 6:
                cout << "Exiting the
program." << endl;
                return 0;


// and...

void displayStack() {
    if (isEmpty()) {
```

```
    cout << "Enter number of max elements
for new stack (max " << maxCap << "): ";
    cin >> i;



    while (true) {
        cout << "\nStack Operations: " <<
endl;
        cout << "1. PUSH, 2. POP, 3. TOP,
4. isEMPTY, 5. DISPLAY, 6. EXIT" << endl;
        cin >> choice;

        switch (choice) {
            case 1:
                push();
                break;
            case 2:
                pop();
                break;
            case 3:
                Top();
                break;
            case 4:
                cout << isEmpty() << endl;
                break;
            case 5:
                displayStack();
                break;
            case 6:
                cout << "Exiting the
program." << endl;
                return 0;
            default:
                cout << "Invalid Choice."
<< endl;
                break;
        }
    }

    return 0;
}

bool isEmpty() {
    return top == -1;
}

void push() {
    if (top == i - 1) {
        cout << "Stack Overflow." << endl;
        return;
    }
    cout << "New Value: ";
    cin >> newData;
    stack[++top] = newData;
```

```
        cout << "The stack is empty."
<< endl;
        return;
    } else {
        cout << "Stack elements (from
top to bottom): ";
        for (int j = top; j >= 0;
j--) { // Loop from top to bottom
            cout << stack[j];
            if (j > 0) {
                cout << ", ";
            }
        }
        cout << endl;
    }
}
```

EXPLANATION

C/C++

```
     case 5:
                displayStack();
                break;
            case 6:
                cout << "Exiting the
program." << endl;
                return 0;

//in order for the stacks to be
displayed, the user would have to
exit the program


void displayStack() { //this is a
function definition
    if (isEmpty()) {         cout <<
"The stack is empty." << endl;
        return;
// this checks if whether the stack
is empty if it is it will give an
poutput of "The stack is empty

    } else {
        cout << "Stack elements (from
top to bottom): "; // this print if
the stack is not empty and indicated
how the flow of the stack

        for (int j = top; j >= 0;
j--) { // for loop will start with
"j" and will be set at the top, for
```

```
}

void pop() {
    // Check if empty -> if yes, return
error
    if (isEmpty()) {
        cout << "Stack Underflow." << endl;
        return;
    }
    cout << "Popping: " << stack[top] <<
endl;
    top--; // Decrement top value from
stack
}

void Top() {
    if (isEmpty()) {
        cout << "Stack is Empty." << endl;
        return;
    }
    cout << "The element on the top of the
stack is " << stack[top] << endl;
}

void displayStack() {
    if (isEmpty()) {
        cout << "The stack is empty." <<
endl;
        return;
    } else {
        cout << "Stack elements (from top
to bottom): ";
        for (int j = top; j >= 0; j--) { //
Loop from top to bottom
            cout << stack[j];
            if (j > 0) {
                cout << ", ";
            }
        }
        cout << endl;
    }
}
```

```
exampl if the first input is 3 thenj
will be 3 and there is a decrements j
until it reaches 0 wherein it
indicated the stacks will begin from
top to bottom
            cout << stack[j];  // it
will print the current element within
the "j"
            if (j > 0) {
                cout << ", "; // used
to separate the following elements
            }
        }
        cout << endl;
    }
}
```

```
Enter number of max elements for new stack (max 100): 4

Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEMPTY, 5. DISPLAY, 6. EXIT
1
New Value: 1

Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEMPTY, 5. DISPLAY, 6. EXIT
1
New Value: 2

Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEMPTY, 5. DISPLAY, 6. EXIT
1
New Value: 3

Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEMPTY, 5. DISPLAY, 6. EXIT
1
New Value: 5

Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEMPTY, 5. DISPLAY, 6. EXIT
5
Stack elements (from top to bottom): 5, 3, 2, 1

Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEMPTY, 5. DISPLAY, 6. EXIT
6
Exiting the program.
```

TABLE 4.2 ILO B-1

OUTPUT	OBSERVATION/EXPLANATION

C/C++

```
#include <iostream>

class Node {
public:
    int data;
    Node *next;
};

Node *head = NULL;

void push(int newData) {
    Node *newNode = new Node;
    newNode->data = newData;
    newNode->next = head;
    head = newNode;
}

int pop() {
    if (head == NULL) {
        std::cout << "Stack Underflow."
<< std::endl;
        return -1;
    } else {
```

C/C++

```
//THE NEWLY ADDED CODE

void displayStack() {
    if (head == NULL) {
        std::cout << "There is nothing
to display." << std::endl;
        return;
    }

    Node *current = head;
    std::cout << "Stack elements: ";

    while (current != NULL) {
        std::cout << current->data;
        current = current->next;

        if (current != NULL) {
            std::cout << ",";
        }
    }

    std::cout << std::endl;
}
```

```
        int tempVal = head->data;
        Node *temp = head;
        head = head->next;
        delete temp;
        return tempVal;
    }
}

void Top() {
    if (head == NULL) {
        std::cout << "Stack is Empty."
<< std::endl;
    } else {
        std::cout << "Top of Stack: " <<
head->data << std::endl;
    }
}

void displayStack() {
    if (head == NULL) {
        std::cout << "There is nothing
to display." << std::endl;
        return;
    }

    Node *current = head;
    std::cout << "Stack elements: ";

    while (current != NULL) {
        std::cout << current->data;
        current = current->next;

        if (current != NULL) {
            std::cout << ",";
        }
    }

    std::cout << std::endl;
}

int main() {
    push(1);
    std::cout << "After the first PUSH,
top of stack is: ";
    Top();
    displayStack();

    push(5);
    std::cout << "After the second PUSH,
top of stack is: ";
    Top();
    displayStack();

    pop();
```

EXPLANATION

C/C++

```
void displayStack() { // this is the
function definition

    if (head == NULL) { // this line
implies that if the start of the stack
which is the head is NULL or there is
nothing in it, it would mean that the
stack is empty

        std::cout << "There is nothing
to display." << std::endl; //If the
stack is indeed empty, it will print
out, "There is nothing to display"
        return;
    }



Node *current = head;  // the current
will go through everything that is in
the stack one after another, otherwise
known as traversing
    std::cout << "Stack elements: ";
//this is for printing the output

    while (current != NULL) {
        std::cout << current->data;
        current = current->next; //this
implies that if the current is not NULL
or empty it will loop through each node
and the data will be stored to the
console afterwards the current will be
updated to the next node in the stack

        if (current != NULL) {
            std::cout << ",";
        } // if the current is not
empty, it will place a ",: between
inputs.
    }

    std::cout << std::endl;
}
```

```
    std::cout << "After the first POP
 operation, top of stack is: ";
    Top();
    displayStack();

    pop();
    std::cout << "After the second POP
 operation, top of stack is: ";
    Top();
    displayStack();


    return 0;
}
```

```
After the first PUSH, top of stack is: Top of Stack: 1
Stack elements: 1
After the second PUSH, top of stack is: Top of Stack: 5
Stack elements: 5,1
After the first POP operation, top of stack is: Top of Stack: 1
Stack elements: 1
After the second POP operation, top of stack is: Stack is Empty.
There is nothing to display.
```

TABLE 4.3 ILO B-2

7. Supplementary Activity

C/C++

```
#include <iostream>
#include <stack>
#include <string>
using namespace std;

int main() {
    string expressions[] = {
        "(A+B)+(C-D)",
        "((A+B)+(C-D)",
        "((A+B)+[C-D])",
        "((A+B)+[C-D}]" // objectives: to check if the bracket, curly braces, and
parentheses matched in the math expression
    };

    // lets first set up the symbols to be used which are: (),[],and {}

    bool isOpen(char ch) { // Function to check if it's an opening symbol
        return (ch == '(' || ch == '{' || ch == '[');
    }

    bool isClose(char ch) { // Function to check if it's a closing symbol
        return (ch == ')' || ch == '}' || ch == ']');
    }
```

```
    // bool is a function which can tell if the value is true of false, the bool was used
in this code so that we will know if the symbols match with one another
    // "||" is an operator known as "OR" used to connection expression into one, one value
must be present in order for the expression to be true
    bool isDoTheyMatch(char open, char close) {
        return (open == '(' && close == ')') ||
               (open == '{' && close == '}') ||
               (open == '[' && close == ']');
    }

    // "&&" is an operator known as "AND", it is a logical operator, it connects two
expressions to one. In this situation, both the open and close values should match in order
for the statement to be true

    bool isItBalanced(const string &expression) { // checking if the expression is balanced
        // const string means CONSTANT STRING, this will ensure there will be no actual
changes in the string that was placed
        stack<char> s; // Create a stack to hold opening symbols
                       // "s" means stacks

        for (char ch : expression) {
            if (isOpen(ch)) {
                s.push(ch);  // This is to Push opening symbols onto the stack
            } else if (isClose(ch)) {
                if (s.empty()) {
                    return false;  // Stack is empty, no opening symbol to match
                }
                if (!isDoTheyMatch(s.top(), ch)) {
                    return false;  // The current closing symbol does not match the top of
the stack
                }
                s.pop();  // Pop the opening symbol if matched
            }
        }

        return s.empty();  // If stack is empty at the end, parentheses are balanced
    }

    // Check each expression and display results
    for (const string &exp : expressions) {
        cout << "Expression: " << exp << endl;
        if (isItBalanced(exp)) { // Call the isItBalanced function to check balance
            cout << "Valid? Yes" << endl;
        } else {
            cout << "Valid? No" << endl;
        }
        cout << endl;
    }

    return 0;
}
```

EXPRESSION	Valid ? (Y/N)	Output (Console Screenshot)	Analysis
(A+B)+(C-D)	y	Expression: (A+B)+(C-D) Valid? Yes	the expression is homogenous, there is no (),{},[] that does not match from one another hence that is why the output confirms that it is valid
((A+B)+(C-D)	n	Expression: ((A+B)+(C-D) Valid? No	the expression is not homogenous, there is a missing “)” at the outer part of the expression hence that’s why the output confirms that the expression is not valid
((A+B)+[C-D])	y	Expression: ((A+B)+[C-D]) Valid? Yes	the expression is homogenous, there is no (),{},[] that does not match from one another either from the inner or outer hence that is why the output confirms that it is valid
((A+B]+[C-D]}	n	Expression: ((A+B)+[C-D}] Valid? No	the expression is not homogenous, the open parenthesis “((“ does not match the closing parenthesis “}] hence that’s why the output confirms that the expression is not valid

8. Conclusion

I learned that there are many ways where you can utilize stacks and maximize their purpose in various ways such as confirming if the expression is correct, displaying stacks output with user input, and identifying the attributes of the stacks. I was able to utilize constructors such as top(), push(), pop(), and more which was very interesting. Stacks can be used in various ways such as keeping data organized and making sure that the entire code is readable and organized

9. Assessment Rubric