15B17CI371 – Data Structures Lab
ODD 2024
Week 2-LAB A
Practice Lab
[CO: C270.1]

1. What is output of the following segment of code? Justify your output by performing a dry run i.e., showcasing each step and change in value for each variable.

```
stackType<int> stack;
int x, y;
x = 4;
y = 0;
stack.push(7);
stack.push(x);
stack.push(x + 5);
y = stack.top();
stack.pop();
stack.push(x + y);
stack.push(y - 2);
stack.push(3);
x = stack.top();
stack.pop();
cout << "x = " << x << endl;
cout << "y = " << y << endl;
while (!stack.isEmptyStack())
{
cout << stack.top() << endl;
stack.pop();
}
```

Dry Run Explanation:

x = 4, y = 0.
Stack: []
stack.push(7):
Stack: [7].
stack.push(x):
Stack: [7, 4].
stack.push(x + 5):
Stack: [7, 4, 9].
y = stack.top():
y = 9.
Stack: [7, 4, 9].
stack.pop():
Stack: [7, 4].

stack.push(x + y):
Stack: [7, 4, 13] (x + y = 4 + 9 = 13).
stack.push(y - 2):
Stack: [7, 4, 13, 7] (y - 2 = 9 - 2 = 7).
stack.push(3):
Stack: [7, 4, 13, 7, 3].
x = stack.top():
x = 3.
Stack: [7, 4, 13, 7, 3].
stack.pop():
Stack: [7, 4, 13, 7].

1. **Output**:

x = 3
y = 9
Print remaining stack:
Outputs: 7 13 4

2. Write a program that uses a stack to print the prime factors of a positive integer in descending order. (You can try to do this using STL also, but for submission purposes DO NOT use STL.)

```
#include <iostream>

using namespace std;



struct Node {

   int data;

   Node* next;

};



class Stack {

private:

   Node* top;

public:

   Stack() : top(nullptr) {}
```

```
void push(int val) {
    Node* n = new Node();
    n->data = val;
    n->next = top;
    top = n;
}

int pop() {
    if (top == nullptr) return -1;
    int val = top->data;
    Node* temp = top;
    top = top->next;
    delete temp;
    return val;
}

bool isEmpty() {
    return top == nullptr;
}

void printStack() {
    while (!isEmpty()) {
        cout << pop() << " ";
```

```
        }
        cout << endl;
    }
};

void printPrimeFactors(int n) {
    Stack stack;
    for (int i = 2; i <= n; i++) {
        while (n % i == 0) {
            stack.push(i);
            n /= i;
        }
    }
    stack.printStack(); // Print factors in descending order
}

int main() {
    int n;
    cout << "Enter a number: ";
    cin >> n;
    printPrimeFactors(n);
    return 0;
}
```

```
Enter a number: 65
13 5

Process returned 0 (0x0)   execution time : 3.197 s
Press any key to continue.
```

3. Write a program to split a stack into two stacks with one containing the bottom half elements and the second the remaining elements; and to combine two stacks into one by placing all elements of the second stack on top of those of the first stack.

```
#include <iostream>

using namespace std;



struct Node {

    int data;

    Node* next;

};



class Stack {

private:

    Node* top;

public:

    Stack() : top(nullptr) {}



    void push(int val) {

        Node* n = new Node();
```

```
    n->data = val;
    n->next = top;
    top = n;
}

int pop() {
    if (top == nullptr) return -1;
    int val = top->data;
    Node* temp = top;
    top = top->next;
    delete temp;
    return val;
}

bool isEmpty() {
    return top == nullptr;
}

void printStack() {
    while (!isEmpty()) {
        cout << pop() << " ";
    }
    cout << endl;
```

```
    }

    void reverseStack() {
        Stack tempStack;
        while (!isEmpty()) {
            tempStack.push(pop());
        }
        while (!tempStack.isEmpty()) {
            push(tempStack.pop());
        }
    }
};

void splitStack(Stack& original, Stack& bottomHalf, Stack& topHalf) {
    original.reverseStack(); // So that we pop from the bottom

    int count = 0;
    while (!original.isEmpty()) {
        if (count % 2 == 0)
            bottomHalf.push(original.pop());
        else
            topHalf.push(original.pop());
        count++;
```

```
    }
}

void combineStacks(Stack& stack1, Stack& stack2) {
    stack2.reverseStack();
    while (!stack2.isEmpty()) {
        stack1.push(stack2.pop());
    }
}

int main() {
    Stack stack1, stack2, bottomHalf, topHalf;
    int n, val;

    cout << "Enter the number of elements in the stack: ";
    cin >> n;
    for (int i = 0; i < n; ++i) {
        cout << "Enter value: ";
        cin >> val;
        stack1.push(val);
    }

    splitStack(stack1, bottomHalf, topHalf);
```

```
    cout << "Bottom half of stack: ";
    bottomHalf.printStack();

    cout << "Top half of stack: ";
    topHalf.printStack();

    combineStacks(stack1, topHalf);
    combineStacks(stack1, bottomHalf);

    cout << "Combined stack: ";
    stack1.printStack();

    return 0;
}
```

```
Enter the number of elements in the stack: 3
Enter value: 1
Enter value: 2
Enter value: 3
Bottom half of stack: 1 3
Top half of stack: 2
Combined stack:

Process returned 0 (0x0)   execution time : 3.850 s
Press any key to continue.
```

4. Write a program to convert a number from decimal notation to a number expressed in a number system whose base (or radix) is a number between 2

and 9. The conversion is performed by repetitious division by the base to which a number is being converted and then taking the remainders of division in the reverse order.

For example, in converting to binary, number 6 requires three such divisions: 6/2 = 3 remainder 0,

3/2 = 1 remainder 1, and finally,

1/2 = 0 remainder 1.

The remainders 0, 1, and 1 are put in the reverse order so that binary equivalent of 6 is equal to 110.

Use stacks/queues to implement the conversion of integer of base 10 to binary

```
#include <iostream>

using namespace std;



struct Node {

    int data;

    Node* next;

};



class Stack {

private:

    Node* top;

public:

    Stack() : top(nullptr) {}



    void push(int val) {

        Node* n = new Node();

        n->data = val;

        n->next = top;
```

```
        top = n;
    }

    int pop() {
        if (top == nullptr) return -1;
        int val = top->data;
        Node* temp = top;
        top = top->next;
        delete temp;
        return val;
    }

    bool isEmpty() {
        return top == nullptr;
    }

    void printStack() {
        while (!isEmpty()) {
            cout << pop();
        }
        cout << endl;
    }
};
```

```
void convertToBase(int num, int base) {
    Stack stack;
    while (num > 0) {
        stack.push(num % base);
        num /= base;
    }
    stack.printStack(); // Print in reverse order
}


int main() {
    int num, base;
    cout << "Enter a decimal number: ";
    cin >> num;
    cout << "Enter the base (2 to 9): ";
    cin >> base;
    convertToBase(num, base);
    return 0;
}
```

5. Write a program to convert (a) given postfix to prefix (b) given prefix to postfix (c) given infix to postfix and further evaluate it to obtain the computed value. Example input: (4 + 9 * 6) - ((8 – 6) / 2 * 4) * 9 / 3

```
#include <iostream>
#include <stack>
#include <string>
```

```
#include <cmath>

using namespace std;

bool isOperator(char c)

{

    return (c == '+' || c == '-' || c == '*' || c == '/' || c == '^');

}

int precedence(char op)

{

    if (op == '+' || op == '-')

        return 1;

    if (op == '*' || op == '/')

        return 2;

    if (op == '^')

        return 3;

    return 0;

}

string postfixToPrefix(string postfix)

{

    stack<string> s;

    for (char c : postfix)

    {

        if (isOperator(c))

        {

            string op2 = s.top();
```

```
            s.pop();
            string op1 = s.top();
            s.pop();
            string temp = c + op1 + op2;
            s.push(temp);
        }
        else
        {
            s.push(string(1, c));
        }
    }
    return s.top();
}
string prefixToPostfix(string prefix)
{
    stack<string> s;
    for (int i = prefix.length() - 1; i >= 0; i--)
    {
        char c = prefix[i];
        if (isOperator(c))
        {
            string op1 = s.top();
            s.pop();
            string op2 = s.top();
```

```
            s.pop();
            string temp = op1 + op2 + c;
            s.push(temp);
        }
        else
        {
            s.push(string(1, c));
        }
    }
    return s.top();
}
string infixToPostfix(string infix)
{
    stack<char> s;
    string postfix = "";
    for (char c : infix)
    {
        if (isdigit(c))
        {
            postfix += c;
        }
        else if (c == '(')
        {
            s.push(c);
```

```
    }
    else if (c == ')')
    {
        while (!s.empty() && s.top() != '(')
        {
            postfix += s.top();
            s.pop();
        }
        s.pop();
    }
    else if (isOperator(c))
    {
        while (!s.empty() && precedence(s.top()) >= precedence(c))
        {
            postfix += s.top();
            s.pop();
        }
        s.push(c);
    }
}
while (!s.empty())
{
    postfix += s.top();
    s.pop();
```

```
    }
    return postfix;
}
int evaluatePostfix(string postfix)
{
    stack<int> s;
    for (char c : postfix)
    {
        if (isdigit(c))
        {
            s.push(c - '0');
        }
        else if (isOperator(c))
        {
            int op2 = s.top();
            s.pop();
            int op1 = s.top();
            s.pop();
            switch (c)
            {
            case '+':
                s.push(op1 + op2);
                break;
            case '-':
```

```
            s.push(op1 - op2);
            break;
        case '*':
            s.push(op1 * op2);
            break;
        case '/':
            s.push(op1 / op2);
            break;
        case '^':
            s.push(pow(op1, op2));
            break;
        }
    }
  }
  return s.top();
}
int main()
{
  string infix = "(4+9*6)-((8-6)/2*4)*9/3";
  cout << "Infix Expression: " << infix << endl;
  string postfix = infixToPostfix(infix);
  cout << "Postfix Expression: " << postfix << endl;
  int result = evaluatePostfix(postfix);
  cout << "Evaluated Result: " << result << endl;
```

```
    string prefix = postfixToPrefix(postfix);

    cout << "Prefix Expression (from Postfix): " << prefix << endl;

    string newPostfix = prefixToPostfix(prefix);

    cout << "Postfix Expression (from Prefix): " << newPostfix << endl;

    return 0;

}
```

```
Infix Expression: (4+9*6)-((8-6)/2*4)*9/3
Postfix Expression: 496*+86-2/4*9*3/-
Evaluated Result: 46
Prefix Expression (from Postfix): -+4*96/**/-862493
Postfix Expression (from Prefix): 496*+86-2/4*9*3/-

Process returned 0 (0x0)   execution time : 0.124 s
Press any key to continue.
```

6. Write a program to check for balancing symbols (parentheses forms) in the following languages: (), [], {}

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

bool isBalanced(const string& expr) {
    stack<char> s;
    for (char ch : expr) {
        if (ch == '(' || ch == '[' || ch == '{') {
            s.push(ch);
        } else if (ch == ')' || ch == ']' || ch == '}') {
            if (s.empty()) return false;
            char top = s.top();
            s.pop();
            if ((ch == ')' && top != '(') ||
                (ch == ']' && top != '[') ||
                (ch == '}' && top != '{')) {
                return false;
            }
        }
    }
    return s.empty();
```

```
}

int main() {
    string expr;
    cout << "Enter the expression: ";
    cin >> expr;

    if (isBalanced(expr)) {
        cout << "The expression is balanced." << endl;
    } else {
        cout << "The expression is not balanced." << endl;
    }

    return 0;
}
```

```
Enter the expression: gad[]
The expression is balanced.

Process returned 0 (0x0)   execution time : 4.250 s
Press any key to continue.
```

7. Write a program to compress a given text by removing whitespaces and replacing continuously repeated character by character followed by no. of time, it is repeated.Use queue data structure.

Example:

Input: asd ddfghjdff kj

Output: asd3fghjdf2kj #include <iostream>

#include <queue>

#include <cctype>

using namespace std;

```
void compressText(const string& text) {

    queue<char> q;

    string result;


    for (char ch : text) {

        if (!isspace(ch)) {
```

```
            q.push(ch);
        }
    }

    while (!q.empty()) {
        char current = q.front();
        q.pop();
        int count = 1;

        while (!q.empty() && q.front() == current) {
            q.pop();
            count++;
        }

        result += current;
        if (count > 1) {
            result += to_string(count);
        }
    }

    cout << "Compressed text: " << result << endl;
}

int main() {
    string text;
    cout << "Enter the text: ";
    getline(cin, text);

    compressText(text);
```

```
    return 0;
}
```

```
Enter the text: asjhfkjeh nkaf
Compressed text: asjhfkjehnkaf

Process returned 0 (0x0)   execution time : 2.634 s
Press any key to continue.
```

8. Write the definition of the function 'moveNthFront' that takes as a parameter a positive integer, n. The function moves the nth element of the queue to the front. The order of the remaining elements remains unchanged. example

Input queue = {5, 11, 34, 67, 43, 55} and n = 3.

After a call to the function moveNthFront,

Output queue = {34, 5, 11, 67, 43, 55}.

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

void moveNthFront(queue<int>& q, int n) {
    int size = q.size();
    if (n > size) return;

    queue<int> tempQueue;
    for (int i = 1; i <= size; ++i) {
        int front = q.front();
        q.pop();
        if (i == n) {
```

```
            tempQueue.push(front);
        } else {
            tempQueue.push(front);
        }
    }

    for (int i = 1; i < n; ++i) {
        int front = tempQueue.front();
        tempQueue.pop();
        q.push(front);
    }

    int nthElement = tempQueue.front();
    tempQueue.pop();
    q.push(nthElement);

    while (!tempQueue.empty()) {
        q.push(tempQueue.front());
        tempQueue.pop();
    }
}

void displayQueue(queue<int> q) {
    while (!q.empty()) {
        cout << q.front() << " ";
        q.pop();
    }
    cout << endl;
}
```

```
int main() {
    queue<int> q;
    int n, value;

    cout << "Enter number of elements in queue: ";
    int count;
    cin >> count;
    cout << "Enter the elements: ";
    for (int i = 0; i < count; ++i) {
        cin >> value;
        q.push(value);
    }

    cout << "Enter the position of element to move to the front: ";
    cin >> n;

    moveNthFront(q, n);

    cout << "Modified queue: ";
    displayQueue(q);

    return 0;
```

}

```
Enter number of elements in queue: 5
Enter the elements: 2
3
4
1
5
Enter the position of element to move to the front: 3
Modified queue: 2 3 4 1 5

Process returned 0 (0x0)   execution time : 19.827 s
Press any key to continue.
```

9. Write a program that reads a line of text, changes each uppercase letter to lowercase, and places each letter both in a queue and onto a stack. The program should then verify whether the line of text is a palindrome (a set of letters or numbers that is the same whether read forward or backward).

```
#include <iostream>

#include <queue>

#include <stack>

#include <cctype>

using namespace std;


bool isPalindrome(const string& text) {

   queue<char> q;

   stack<char> s;


   for (char ch : text) {

      if (isalpha(ch)) {

         char lowerCh = tolower(ch);

         q.push(lowerCh);

         s.push(lowerCh);

      }
```

```
    }

    while (!q.empty()) {
        if (q.front() != s.top()) {
            return false;
        }
        q.pop();
        s.pop();
    }

    return true;
}

int main() {
    string text;
    cout << "Enter the text: ";
    getline(cin, text);

    if (isPalindrome(text)) {
        cout << "The text is a palindrome." << endl;
    } else {
        cout << "The text is not a palindrome." << endl;
    }

    return 0;
}
```

```
Enter the text: hello
The text is not a palindrome.

Process returned 0 (0x0)   execution time : 6.712 s
Press any key to continue.
```

10. A string of characters is given.

A scientist is interested in a very typical pattern. He wishes to reverse all the characters which lies inside 2 substrings namely S1, and S2. S1 is the string of any length but starts from X and ends with Y. S2, starts from Y and ends with X.

Example:

Input: “ABXNNYPEROYABCDCXT

Output: OREP.

Write a program to solve this problem

```
#include <iostream>

#include <string>

using namespace std;


void reverseSubstring(string& str, size_t start, size_t end) {

   while (start < end) {

      char temp = str[start];

      str[start] = str[end];

      str[end] = temp;

      ++start;

      --end;

   }

}


string reverseInsideSubstrings(const string& str) {
```

```
    size_t start1 = str.find('X');
    size_t end1 = str.find('Y', start1 + 1);
    size_t start2 = str.find('Y', end1 + 1);
    size_t end2 = str.find('X', start2 + 1);

    if (start1 == string::npos || end1 == string::npos ||
        start2 == string::npos || end2 == string::npos || start1 > end1 || start2 > end2) {
        return ""; // Invalid pattern
    }

    string result = str;

    reverseSubstring(result, start1 + 1, end1 - 1);
    reverseSubstring(result, start2 + 1, end2 - 1);
    string finalResult;
    finalResult += result.substr(start1 + 1, end1 - start1 - 1);
    finalResult += result.substr(start2 + 1, end2 - start2 - 1);

    return finalResult;
}

int main() {
    string str;
    cout << "Enter the string: ";
    getline(cin, str);

    string result = reverseInsideSubstrings(str);

    cout << "Result: " << result << endl;
```

```
    return 0;
}
```

```
Enter the string: ABXNNYPEROYABCDCXT
Result: NNCDCBA

Process returned 0 (0x0)   execution time : 1.120 s
Press any key to continue.
```