**Kathmandu University**

**Department of Computer Science and Engineering**

**Dhulikhel, Kavre**



**A Report on**

**COMP 202: Data Structures and Algorithms**

**Mini Project**

**Submitted by:**

Prashanna Mani Paudel (37)

Gaurav Rizal (57)

**Submitted to:**

Dr. Rajani Chulyadyo

Department of Computer Science and Engineering

**Submission Date:**

January 18, 2019

**Task:**  To implement a stack using a circularly linked list with operations push, pop, top, isFull, isEmpty and find the time complexities of these operation.

**Implementation:**

Algorithm

* isEmpty

Input: Stack(top)

Output: true if empty, false if not

Steps:

1. If Stack == NULL
2. Return true
3. Else
4. Return false
5. Endif

* Push

Input: Stack (top), data to be pushed

Output: Returned Stack(top)

Steps:

1. Create a new node, newNode.
2. newNode->info = data
3. If stack is empty:
4. newNode->next = newNode
5. Stack = newNode
6. Else:
7. newNode->next = stack->next
8. Stack->next = newNode
9. Endif
10. Return Stack

* Top

Input: Stack(top)

Output: Top of Stack

Steps:

1. If Stack is empty:
2. “Stack Underflow”
3. Else:
4. print stack->next->info
5. Endif

* Pop

Input: Stack(top)

Output: Returned Stack(top)

Steps:

1. If Stack is empty:
2. “Stack Underflow”
3. Else:
4. top = Stack->next
5. data = top->info
6. Stack->next = top->next
7. If top == Stack:
8. Stack == NULL
9. Endif
10. Endif
11. Return Stack

* Display

Input: Stack(top)

Output: Contents of Stack

Steps:

1. If Stack is empty:
2. “Stack Underflow”
3. Else:
4. Create a new node, newNode
5. newNode = Stack
6. while (newNode->next != Stack):
7. newNode = newNode->next
8. print newNode->info
9. End while
10. newNode = newNode->next
11. print newNode->info

**C++ Source Codes**

The source codes of the program are submitted in the files labeled as linkedlist.h and linked\_list.cpp.

The linkedlist.h contains the declaration of class Stack and class Node which is used to create circularly linked list. The linked\_list.cpp file contains all the operations such as push, pop, isEmpty etc which are required for Stack to be operated. Finally linked\_list.cpp also contains a real-life use of Stack which is reversing a string by the use of Stack.

**Test Program** (Manually done according to the algorithm)

Steps:

1. Push 45
2. Push 6735
3. Push 9856

45

6735

9856

Stack

1. Pop

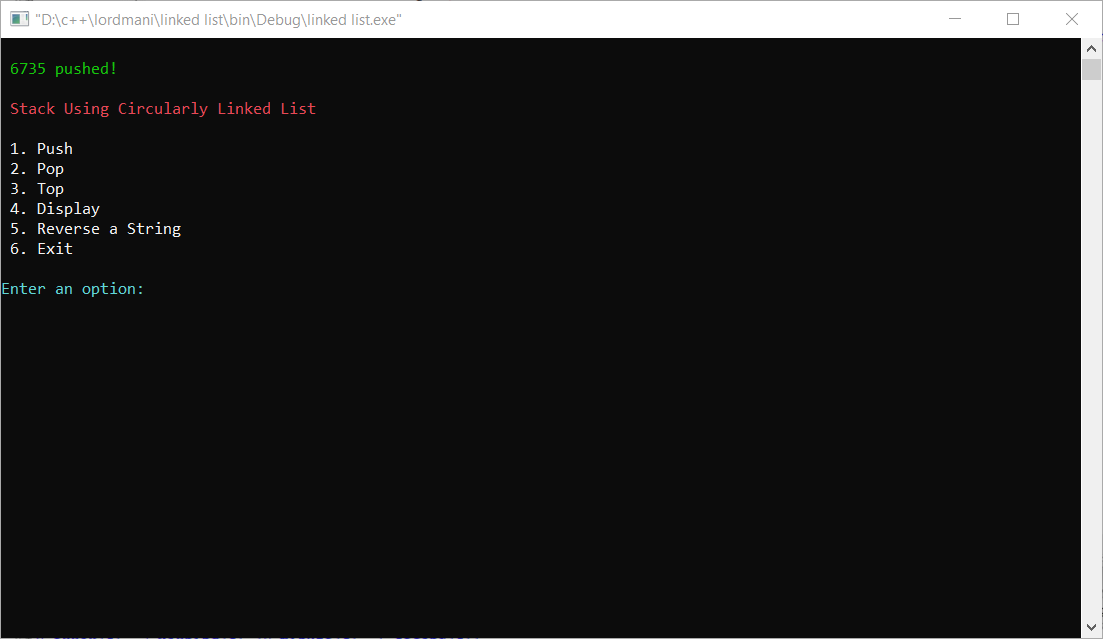
45

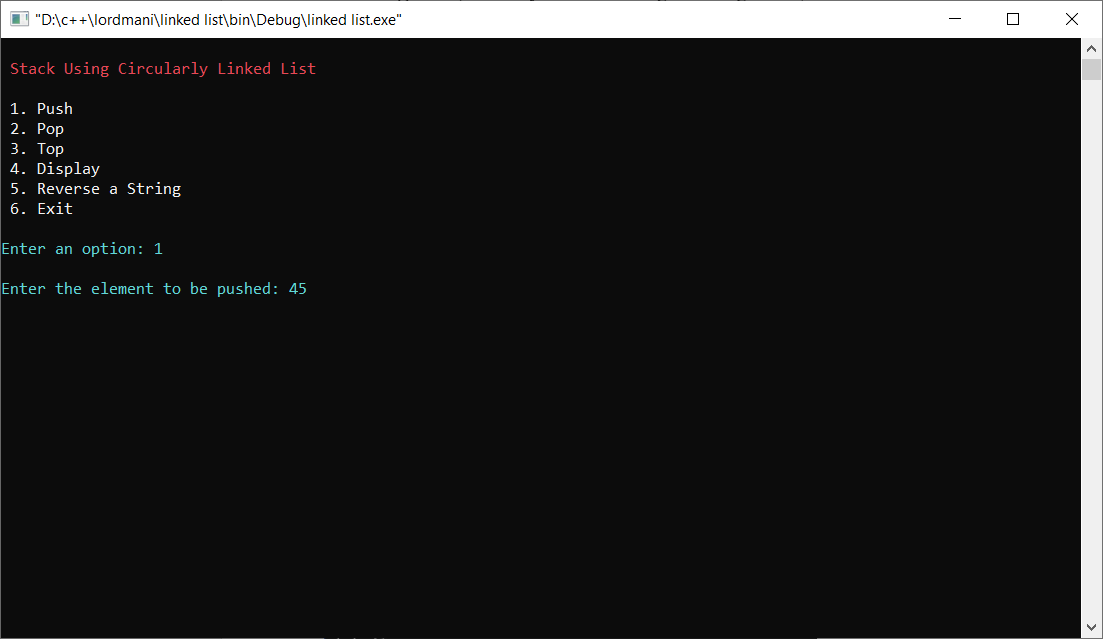
6735

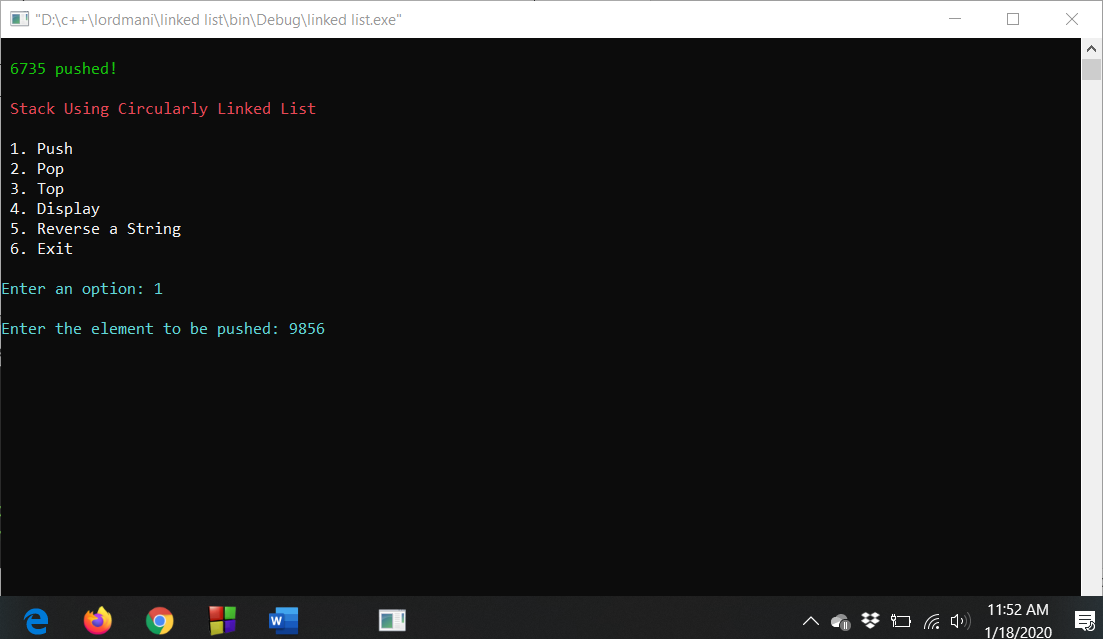
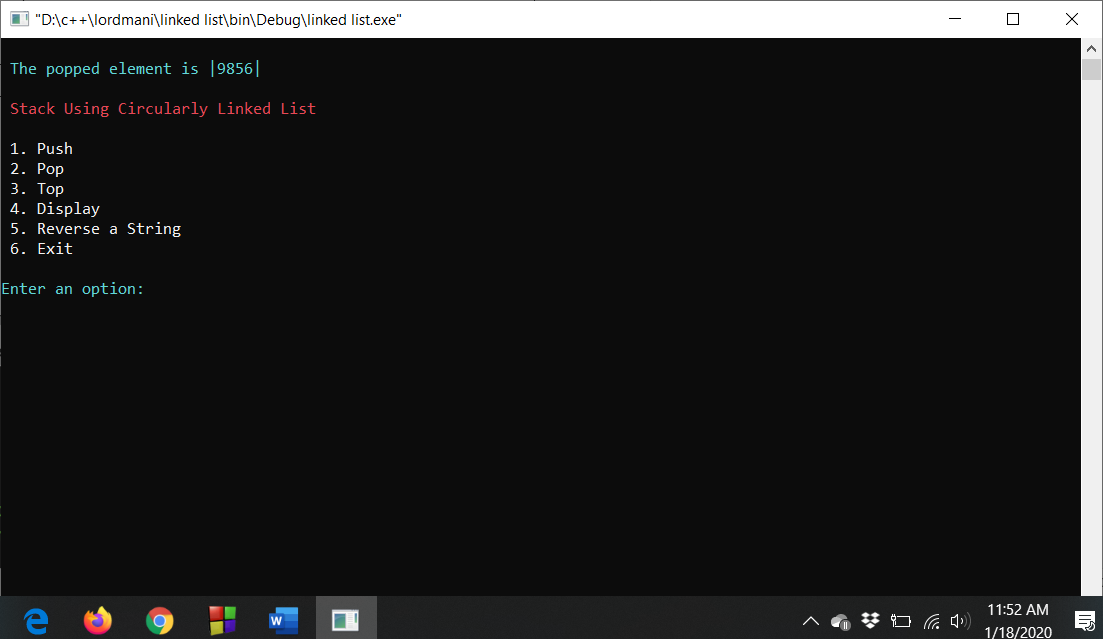
Stack

1. Display

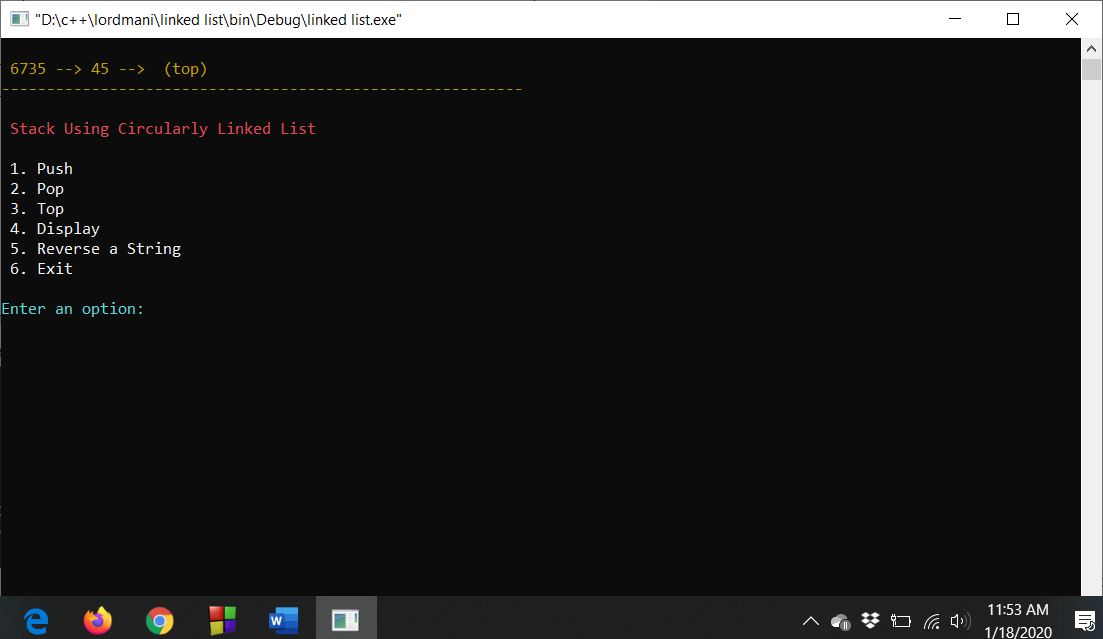
**Expected Output**: 6735 🡪 45

****





**Output from the program:**



Hence, the implementation works correctly.

**Time Complexity of the Program**

**isEmpty**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Program** | **Steps** |
| 1.  2.  3.  4.  5.  6.  7. | bool STACKS::isEmpty(Node\* top\_ele){  if (top\_ele == NULL){  return true;  }else{  return false;  }  } | 0  c1  c1  c2  c2  0  0 |
|  | Total: | r (constant) |

Hence, Time complexity or the total runtime of the program = O(1)

**Top**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Program** | **Steps** |
| 1.  2.  3.  4.  5.  6.  7.  8.  9. | void STACKS::top(Node\* top\_ele){  if (isEmpty(top\_ele)){  SetConsoleTextAttribute(color, 12);  std::cout << "\n Stack underflow \n";  }else{  SetConsoleTextAttribute(color, 10);  std::cout << "\n The top element is " << top\_ele->next->info << std::endl;  }  } | 0  c1  c1  c1  c2  c2  c2  0  0 |
|  | Total: | r (constant) |

Hence, Time complexity or the total runtime of the program = O(1)

**Push**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Program** | **Steps** |
| 1.  2.  3.  4.  5.  6.  7.  8.  9.  10.  11.  12.  13.  14.  15. | Node\* STACKS::push(Node\* top\_ele, std::string data){  Node\* newNode = new Node();  newNode->info = data;  if (isEmpty(top\_ele)){  newNode->next = newNode;  top\_ele = newNode;  }else{  newNode->next = top\_ele->next;  top\_ele->next = newNode;  }  SetConsoleTextAttribute(color, 10);  system("cls");  std::cout << "\n " << data << " pushed! \n" ;  return top\_ele;  } | 0  1  1  c1  c1  c1  1  c2  c2  0  1  1  1  1 |
|  | Total: | r (constant) |

Hence, Time complexity or the total runtime of the program = O(1)

**Pop**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Program** | **Steps** |
| 1.  2.  3.  4.  5.  6.  7.  8.  9.  10.  11.  12.  13.  14.  15.  16.  17.  18.  19.  20. | Node\* STACKS::pop(Node\* top\_ele){  if (isEmpty(top\_ele)){  system("cls");  SetConsoleTextAttribute(color, 12);  std::cout << "\n Stack underflow. Cannot pop. \n" << std::endl;  }else if (top\_ele->next == top\_ele){  system("cls");  SetConsoleTextAttribute(color, 11);  std::cout << "\n The popped element is |" << top\_ele->info << "|" << "\n";  top\_ele == NULL;  }else{  Node\* temp = top\_ele->next;  std::string data = temp->info;  top\_ele->next = temp->next;  system("cls");  SetConsoleTextAttribute(color, 11);  std::cout << "\n The popped element is |" << data << "|" << "\n";  }  return top\_ele;  } | 0  c1  c1  c1  c1  c2  c2  c2  c2  c2  c3  c3  c3  c3  c3  c3  c3  0  1  0 |
|  | Total: | r(constant) |

Hence, Time complexity or the total runtime of the program = O(1)

Here, c1 = number of steps for first condition, c2 = number of steps for second condition , c3 = number of steps for third condition and r is some constants. Each operation is executed only once for each character (e.g. ‘a’) hence the time complexity of each operation is O(1).

**Screenshots:**

Screenshots of operations of the program:

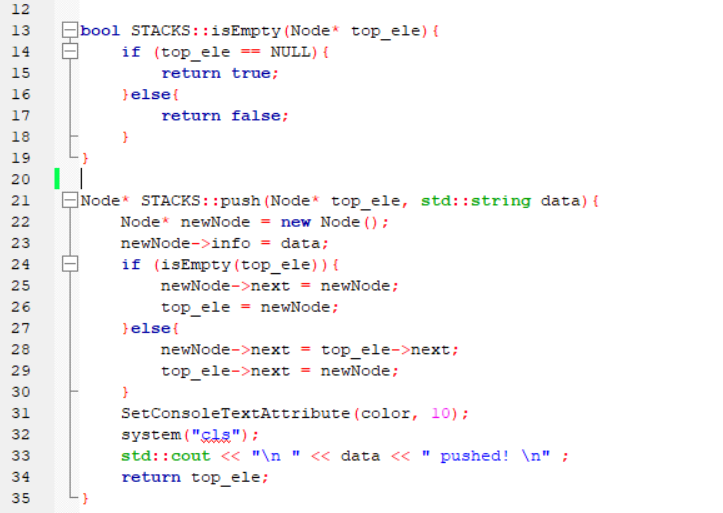


Fig: isEmpty and push operations

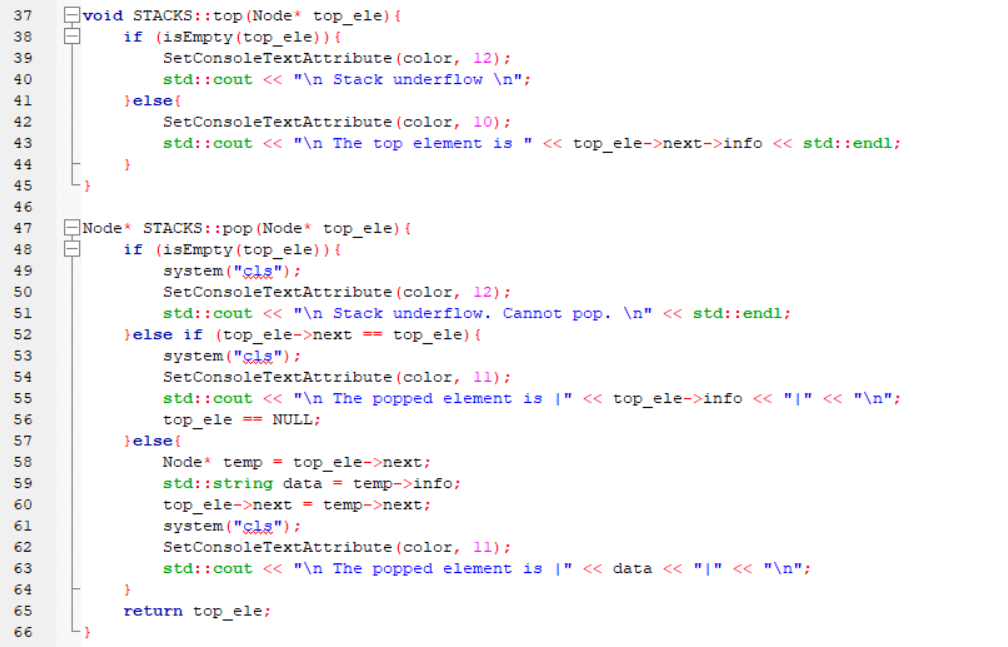


Fig: pop and top operations