

**SEMESTER PROJECT REPORT ON**

**UNDO/REDO OPERATION USING STACK**

**SUBMITTED BY:**

ROLL # 17CS23(G.L)

ROLL # 17CS35

ROLL # 17CS39

ROLL # 17CS73

ROLL # 17CS79

**SUBMITTED TO:**

Dr. Sammer Zai

**DATED: 28/AUG/2019**

Mehran University of Engineering and Technology Jamshoro

Department of Computer Systems Engineering

**Objective**

* To implement **UNDO/REDO** operation using the basic concept of stack known as **PUSH** and **POP.**

**Introduction:**

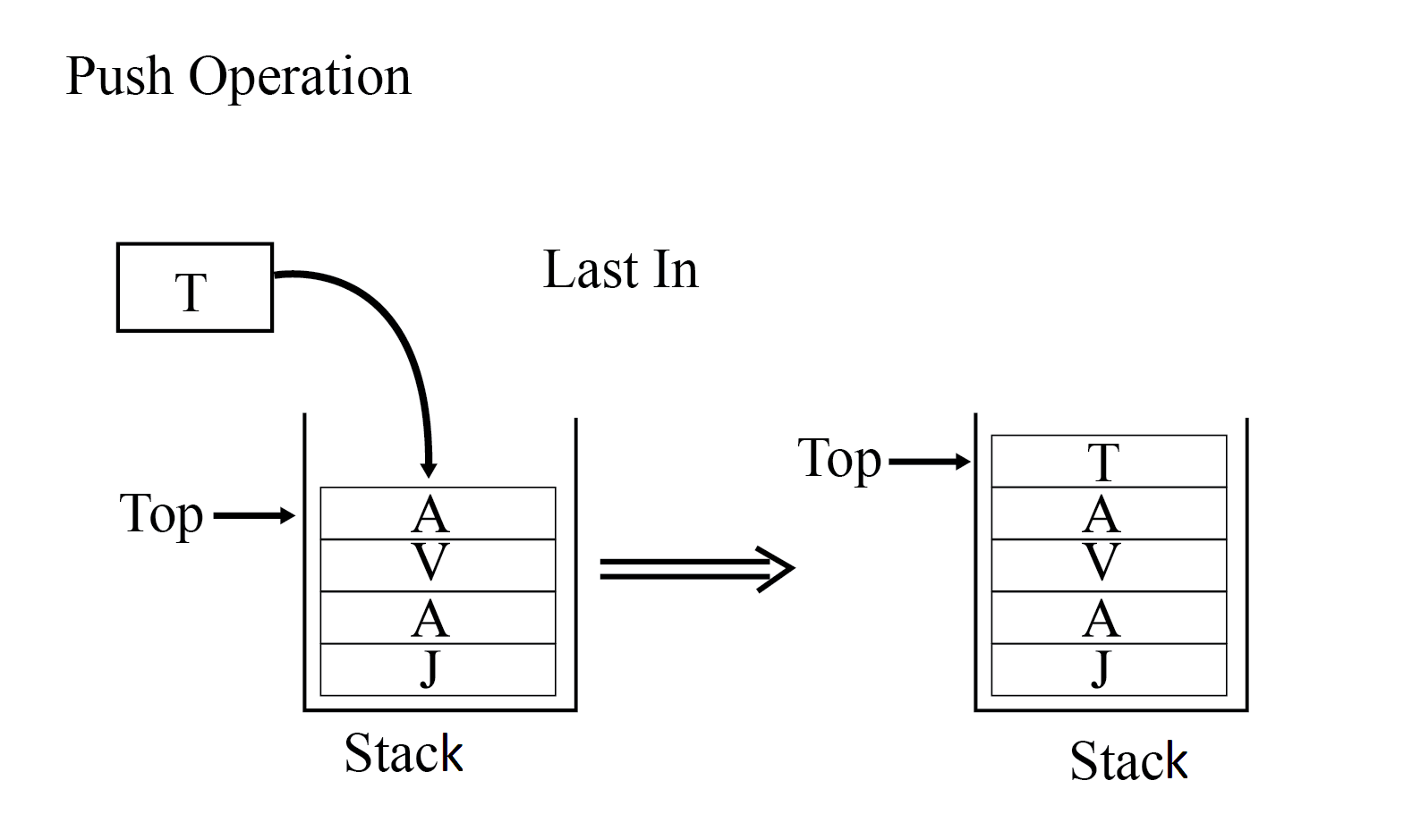
A stack is an **Abstract Data Type** (ADT), commonly used in most programming languages. It is named stack as it behaves like a real-world stack, for example – a deck of cards or a pile of plates, etc.

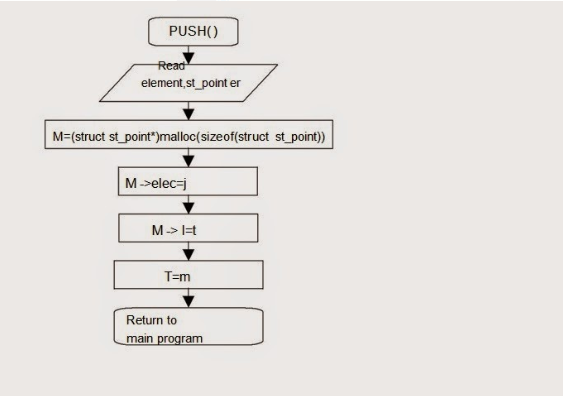
A real-world stack allows operations at one end only. For example, we can place or remove a card or plate from the top of the stack only. Likewise, Stack ADT allows all data operations at one end only. At any given time, we can only access the top element of a stack.

This feature makes it **LIFO** data structure. LIFO stands for Last-in-first-out. Here, the element which is placed (inserted or added) last, is accessed first. In stack terminology, insertion operation is called **PUSH** operation and removal operation is called **POP** operation.

## Push Operation

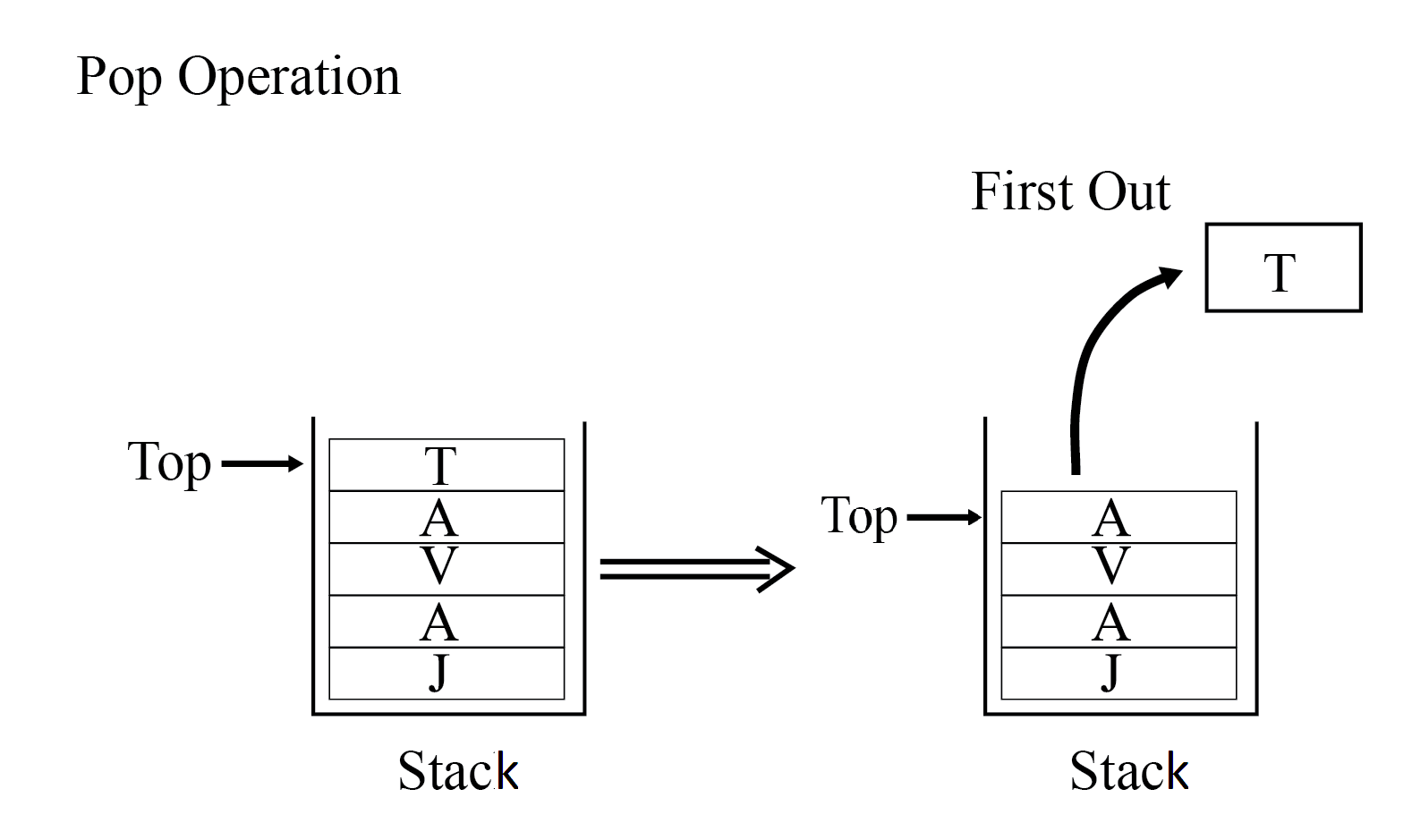
The process of putting a new data element onto stack is known as a Push Operation.

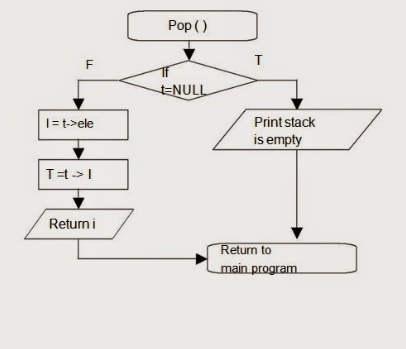




## Pop Operation

Accessing the content while removing it from the stack, is known as a Pop Operation.





**Methodology:**

**General Algorithm of Project:**

1. Display Title
2. Read: menu. [ Project Title Appears]
3. IF menu = $. Display Menu
4. Read: option
5. Do Steps 6 and 7 while choice != N
6. Switch for the value of option:
   1. Case option = ‘C’. [Menu Appears]
   2. Read: choice
      1. Case choice = ‘A’. Perform Addition and Store Result in Stack1
      2. Break
      3. Case choice = ‘S’. Perform Subtraction and Store Result in Stack1
      4. Break
      5. Case choice = ‘M’. Perform Multiplication and Store Result in Stack1
      6. Break
      7. Case choice = ‘D’. Perform Division and Store Result in Stack1
      8. Break
      9. Case choice = ‘R’. Perform Square Root and Store Result in Stack1
      10. Break
      11. Case choice = ‘P’. Perform Power and Store Result in Stack1
      12. Break
      13. Case choice = ‘F’. Perform Factorial and Store Result in Stack1
      14. Break
      15. Otherwise Write: ‘INVALID OPERATION SELECTED’
   3. Case option = ‘U’
      1. Pop out value from Stack1
      2. Push that value in Stack2
   4. Case option = ‘R’
      1. Pop out value from Stack2.
      2. Push that value in Stack1.
   5. Otherwise Write: ‘INVALID OPERATION SELECTED’
7. Read: choice
8. Exit

**Implementation:**

* We have used two arrays as a stack with maximum size of 25.
* First array is used for storing results and second is used for backup.
* When we perform UNDO operation top value of stack1 will pop out and store into stack2.
* When we perform REDO operation value from stack2 will return back to stack1.

**Push1 Function [ To Insert Values into Stack1 for Storing]**

The process of putting a new data element onto stack.

1. IF top1 = MAX. Write: ‘Overflow’.
2. ELSE top1 := top1 +1.
3. Result := stack1[top1]
4. Return.

**Push2 Function [ To Save Values into Stack2 for REDO operation]**

The process of putting a new data element onto stack.

1. IF top2 = MAX. Write: ‘Overflow’.
2. ELSE top2 := top2 +1.
3. Result := stack2[top2]
4. Return.

**Pop1 Function [Delete Values from Stack1 as UNDO operation]**

The process of putting out a new data element from stack.

1. IF top1 = NULL. Write: ‘Underflow’. Return
2. ELSE stack1[top1] := Result
3. top1 := top1 -1.
4. Return.

**Pop2 Function [Delete Values from Stack2 as REDO operation]**

The process of putting out a new data element from stack.

1. IF top2 = NULL. Write: ‘Underflow’. Return
2. ELSE stack2[top2] := Result
3. Top2 := top2 -1.
4. Return.

**HOW UNDO OPERATION IS PERFORMED?**

**Algorithm:**

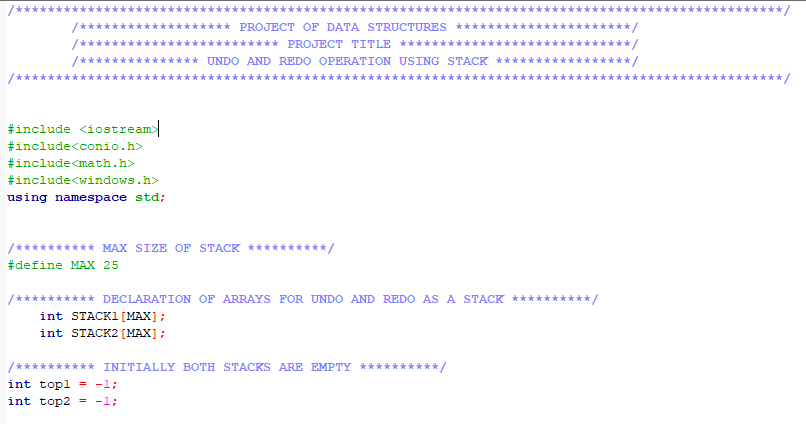
1. Pop value from Stack1
2. Push value onto Stack2
3. Exit

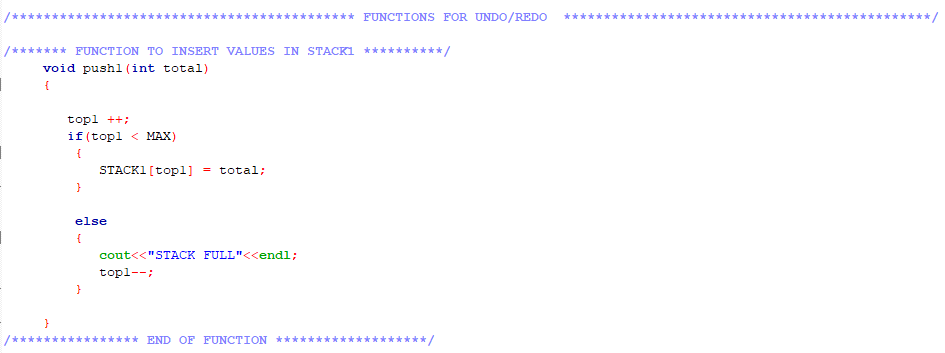
**HOW REDO OPERATION IS PERFORMED?**

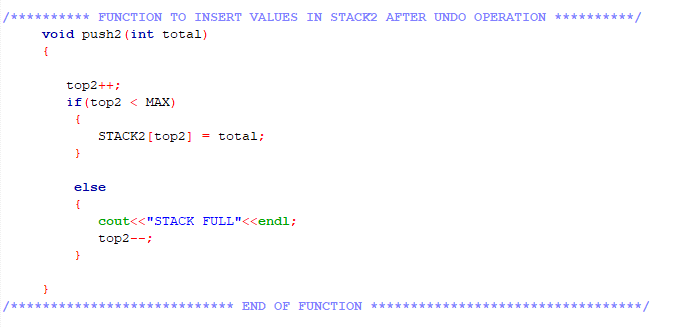
**Algorithm:**

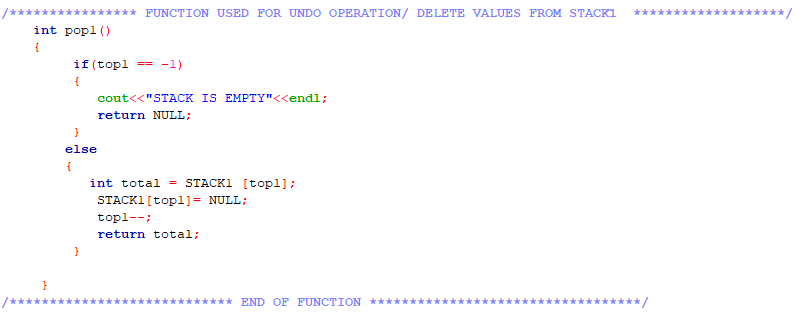
1. Pop value from Stack2
2. Push value onto Stack1
3. Exit

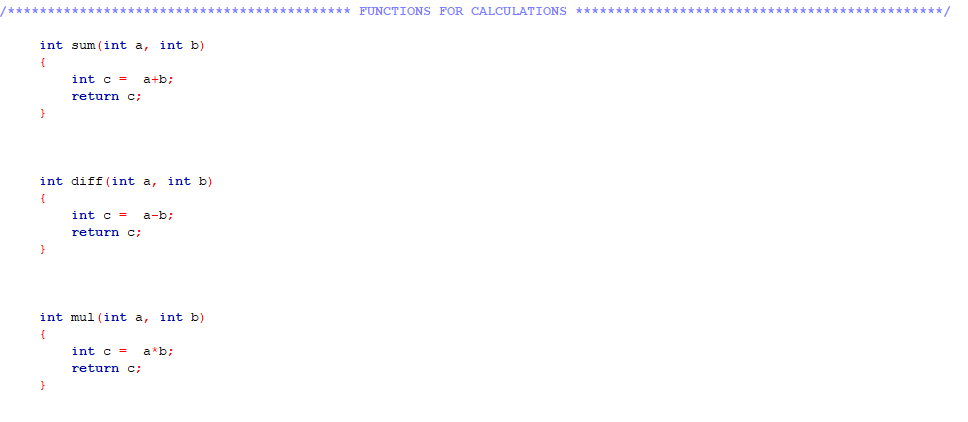
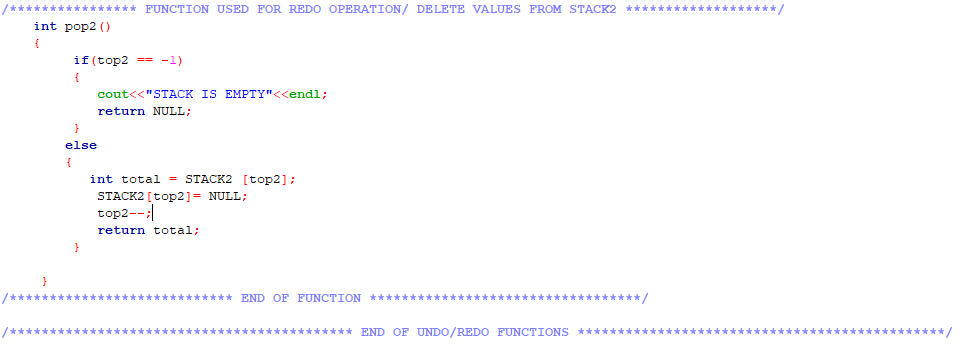
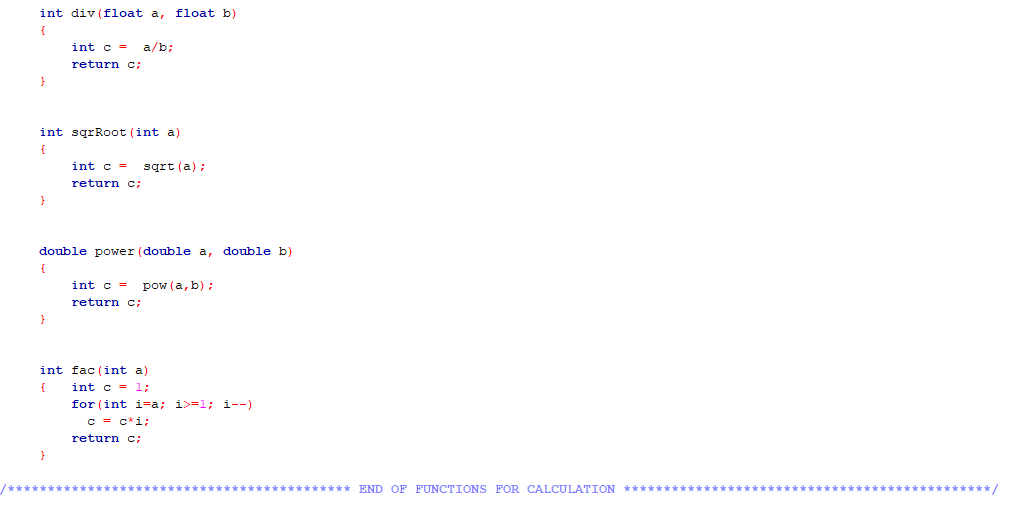
**SOURCE CODE OF OUR PROJECT:**

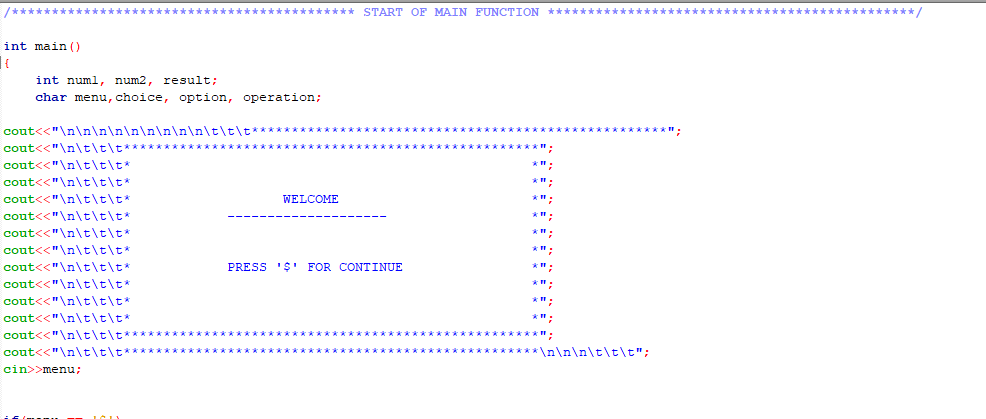


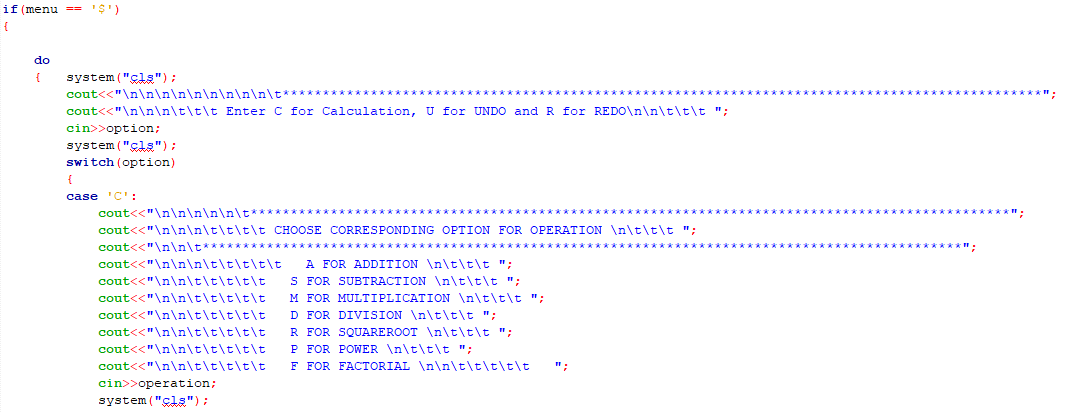


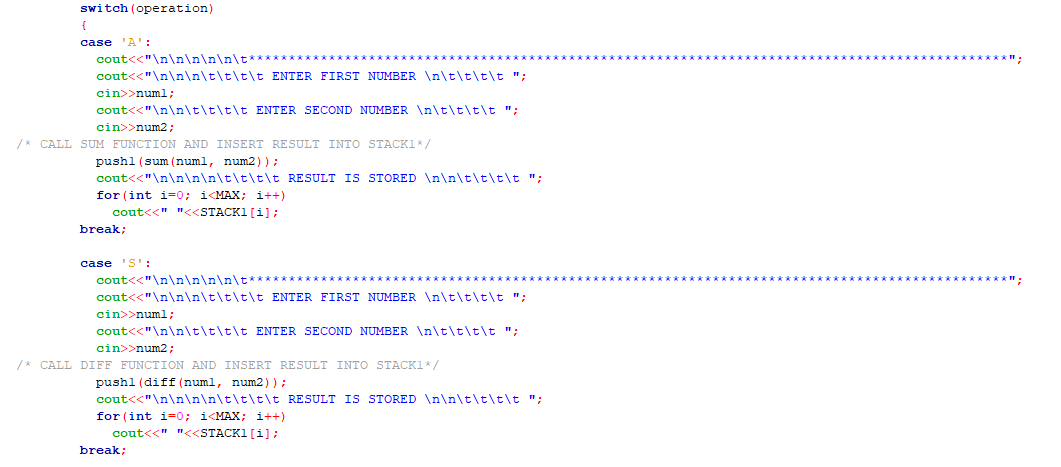


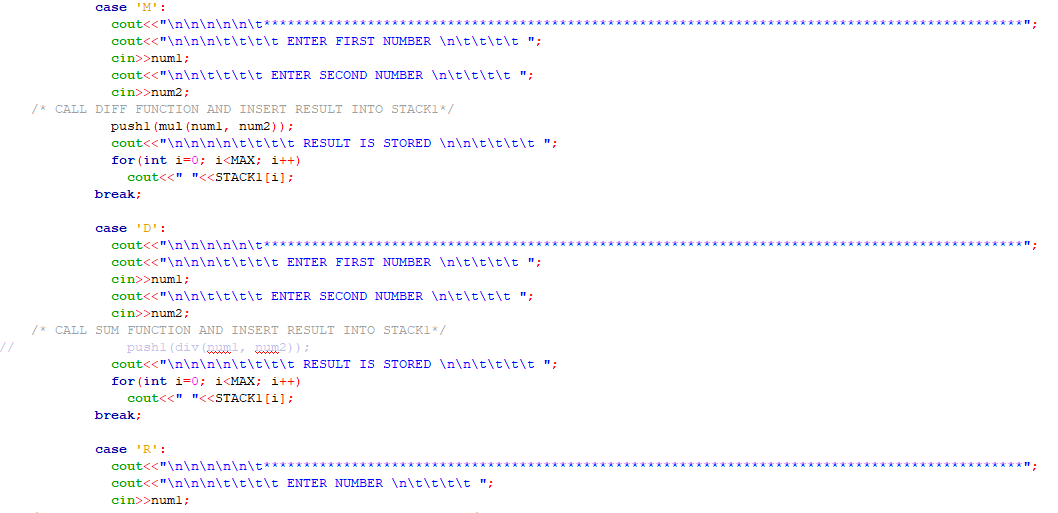


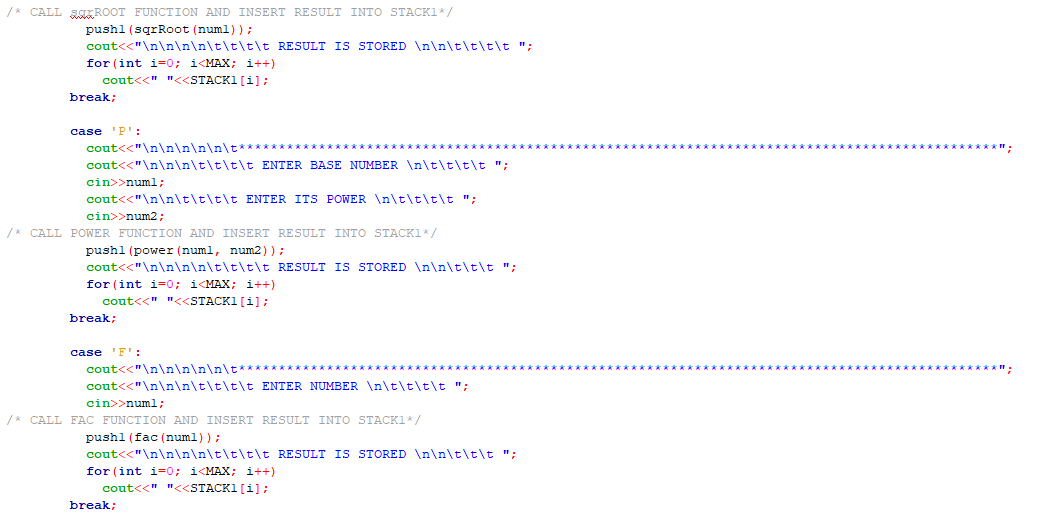
 

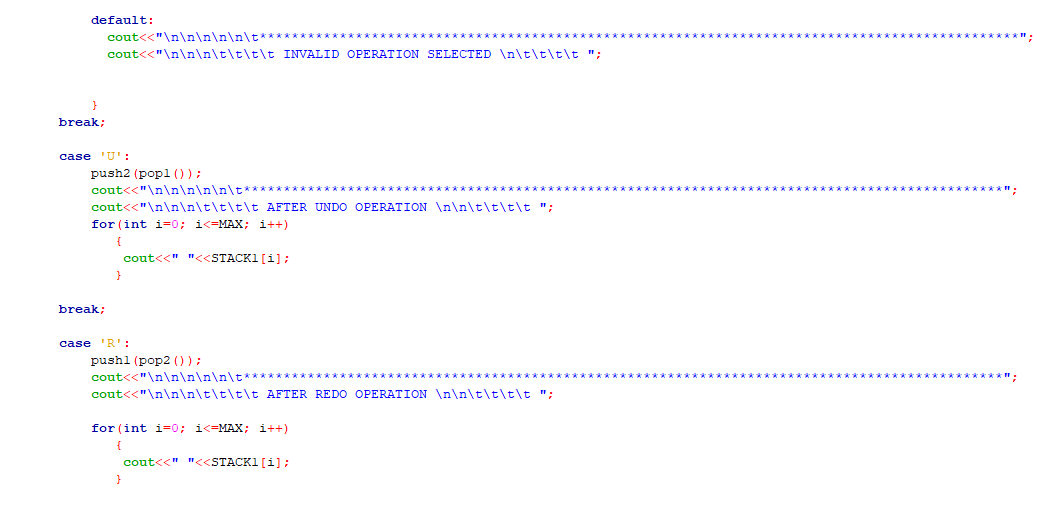


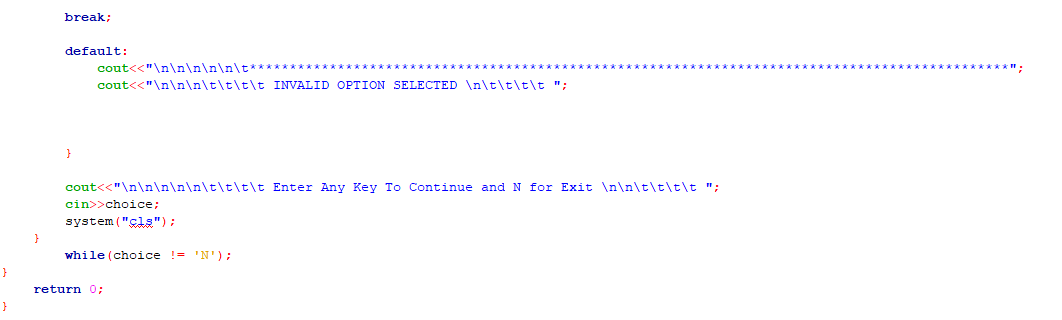






is



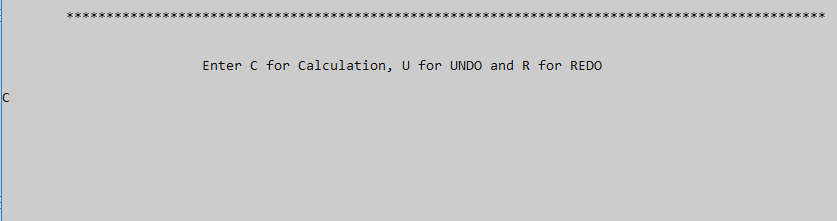


**RESULT OF SOURCE CODE**

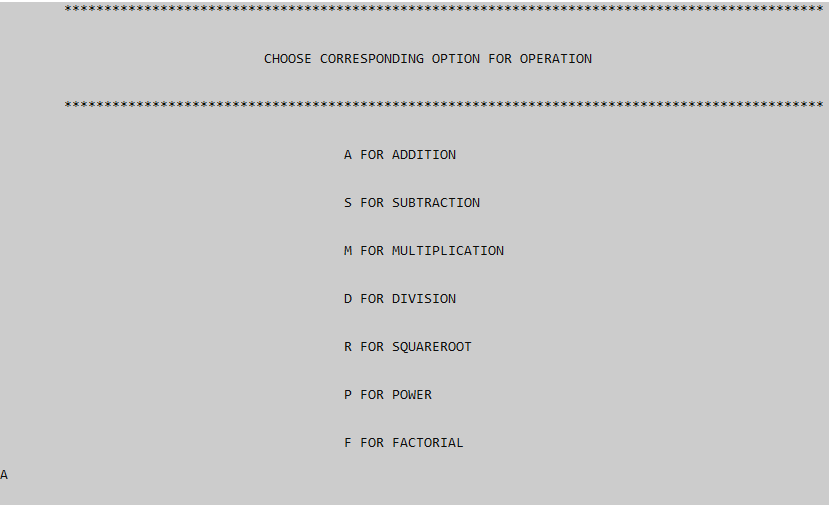
1. Project Title



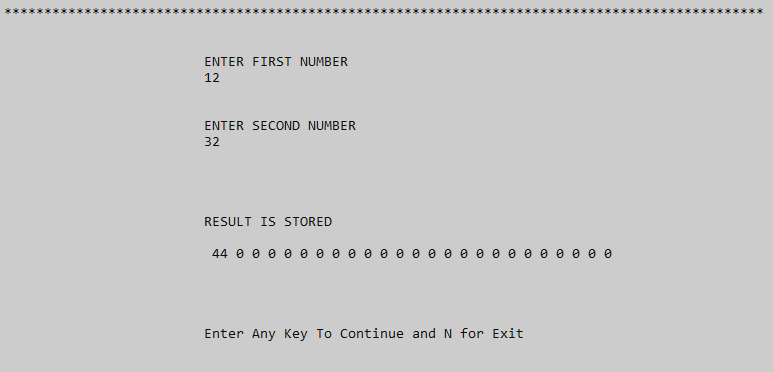
1. Menu Page



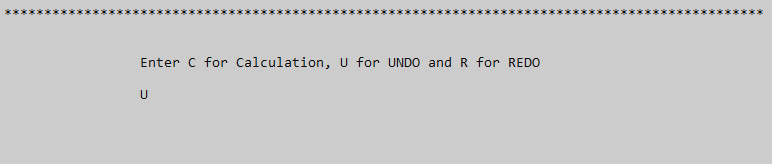
1. Sub Menu Page. Addition is selected.



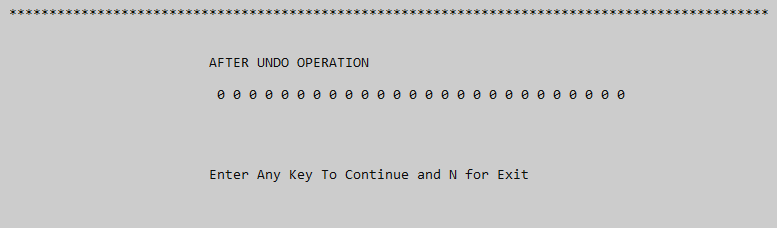
1. Result of Addition is stored in Stack1.



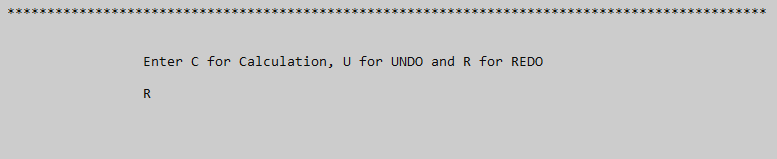
1. Undo Operation Selected



1. Stack After Undo Operation



1. Redo Operation Selected



1. Stack After Redo Operation

