**Lab-1: 19CSE212 Data Structure and Algorithms**

**Implementation of STACK Data Structure as Array**

**ADT of Stack**

**Definition: One side open the other side is closed.**

**Follows the Last in First Out model (LIFO)**

**The top pointer points to the topmost element among all available elements.**

**Operations/Functions:**

1. **PUSH: inserting an element into the stack**
2. **POP: deleting an element from the stack**
3. **PEEP/CHANGE: update/return a value from the stack**

**PUSH Pseudocode**

1. Initially Top = -1
2. PUSH (S, N, Top, x)
3. Begin
4. **if** top = N
5. then print: stack is overflow
6. Exit(1)
7. **else**
8. Top = Top + 1
9. S[top] := item;
10. End

**POP Pseudocode**

1. Initially Top = -1
2. POP (S, N, Top, y)
3. Begin
4. **if** top = -1
5. then print: stack is underflow
6. Exit(1)
7. **else**
8. y := S[Top]
9. Top = Top - 1
10. return y
11. End

**PEEP Pseudocode**

1. Initially Top = -1
2. POP (S, N, Top, I, z)
3. Begin
4. **if** Top-I+1 < 0
5. then print: stack is underflow
6. Exit(1)
7. **else**
8. **Return** S[Top-I+1] :=z
9. End

**C Program to Implement Stack as an Array**

1. #include <stdio.h>
2. **int** stack[100], i, j, choice = 0, n, top = -1 ;
3. **void** push();
4. **void** pop();
5. **void** show();
6. **void** main ()
7. {
8. printf("Enter the number of elements into the stack ");
9. scanf("%d",&n);
10. printf("\*\*\*\*\*\*\*\*\*Stack operations using array\*\*\*\*\*\*\*\*\*");
12. printf("\n----------------------------------------------\n");
13. **while**(choice != 4)
14. {
15. printf("Chose one from the below options...\n");
16. printf("\n1.Push\n2.Pop\n3.Show\n4.Exit");
17. printf("\n Enter your choice \n");
18. scanf("%d",&choice);
19. **switch**(choice)
20. {
21. **case** 1:
22. {
23. push();
24. **break**;
25. }
26. **case** 2:
27. {
28. pop();
29. **break**;
30. }
31. **case** 3:
32. {
33. show();
34. **break**;
35. }
36. **case** 4:
37. {
38. printf("Exiting....");
39. **break**;
40. }
41. **default**:
42. {
43. printf("Please Enter valid choice ");
44. }
45. };
46. }
47. }
49. **void** push ()
50. {
51. **int** val;
52. **if** (top == n )
53. printf("\n Overflow");
54. **else**
55. {
56. printf("Enter the value?");
57. scanf("%d",&val);
58. top = top +1;
59. stack[top] = val;
60. }
61. }
63. **void** pop ()
64. {
65. **if**(top == -1)
66. printf("Underflow");
67. **else**
68. top = top -1;
69. }
70. **void** show()
71. {
72. **for** (i=top;i>=0;i--)
73. {
74. printf("%d\n",stack[i]);
75. }
76. **if**(top == -1)
77. {
78. printf("Stack is empty");
79. }
80. }

**C Program to Implement two Stacks using a Single Array**

1. #include <stdio.h>
2. #define SIZE 30
3. **int** array[SIZE] ;
4. **int** top1 = -1 ;
5. **int** top2 = SIZE ;

//Function to push data into stack1

1. **void** push1 (**int** data)
2. {

// checking the overflow condition

1. **if** (top1 < top2 - 1)
2. {
3. top1++;
4. array[top1] = data;
5. }
6. **else**
7. {
8. printf ("Stack is full");
9. }
10. }

// Function to push data into stack2

1. **void** push2 (**int** data)
2. {
3. // checking overflow condition

1. **if** (top1 < top2 - 1)
2. {
3. top2--;
4. array[top2] = data;
5. }
6. **else**
7. {
8. printf ("Stack is full\n");
9. }
10. }

//Function to pop data from the Stack1

1. **void** pop1 ()
2. {

// Checking the underflow condition

1. **if** (top1 >= 0)
2. {
3. **int** popped\_element = array[top1];
4. top1--;
5. printf ("%d is being popped from Stack 1\n", popped\_element);
6. }
7. **else**
8. {
9. printf ("Stack is Empty \n");
10. }
11. }

// Function to remove the element from the Stack2

1. **void** pop2 ()
2. {

// Checking underflow condition

1. **if** (top2 < SIZE)
2. {
3. **int** popped\_element = array[top2];
4. top2--;
6. printf ("%d is being popped from Stack 1\n", popped\_element);
7. }
8. **else**
9. {
10. printf ("Stack is Empty!\n");
11. }
12. }

//Functions to Print the values of Stack1

1. **void** display\_stack1 ()
2. {
3. **int** i;
4. **for** (i = top1; i >= 0; --i)
5. {
6. printf ("%d ", array[i]);
7. }
8. printf ("\n");
9. }

// Function to print the values of Stack2

1. **void** display\_stack2 ()
2. {
3. **int** i;
4. **for** (i = top2; i < SIZE; ++i)
5. {
6. printf ("%d ", array[i]);
7. }
8. printf ("\n");
9. }
10. **int** main()
11. {
12. **int** ar[SIZE];
13. **int** i;
14. **int** num\_of\_ele;
16. printf ("We can push a total of 30 values\n");
17. //Number of elements pushed in stack 1 is 10
18. //Number of elements pushed in stack 2 is 10
19. // loop to insert the elements into Stack1
20. **for** (i = 1; i <= 10; ++i)
21. {
22. push1(i);
23. printf ("Value Pushed in Stack 1 is %d\n", i);
24. }

// insert the elements into Stack2.

1. **for** (i = 11; i <= 20; ++i)
2. {
3. push2(i);
4. printf ("Value Pushed in Stack 2 is %d\n", i);
5. }

  //Print Both Stacks

1. display\_stack1 ();
2. display\_stack2 ();

//Pushing on Stack Full

1. printf ("Pushing Value into Stack 1 is %d\n", 11);
2. push1 (11);
4. //Popping All Elements from Stack 1
5. num\_of\_ele = top1 + 1;
6. **while** (num\_of\_ele)
7. {
8. pop1 () ;
9. num\_of\_ele;
10. }

// Pop the element From the Empty Stack

1. pop1 ();
3. **return** 0;
4. }

**Assignment:**

**Implement N stacks in an array of size M. Write the pseudocode to show the push and pop operations before implementing the same.**