

BHAVLEEN KAUR
3EIC-2
102155010

LAB ASSIGNMENT 5

Stacks and its Applications

1. Write a menu driven program with 4 options (Push, Pop, Display, and Exit) to demonstrate the working of stacks using arrays.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #define MAX_SIZE 100
4  struct Stack {
5      int items[MAX_SIZE];
6      int top;
7  };
8  void push(struct Stack *s, int value);
9  int pop(struct Stack *s);
10 void display(struct Stack *s);
11 int main() {
12     struct Stack stack;
13     stack.top = -1; // Initialize stack top
14     int choice, value;
15     do {
16         printf("\nStack Operations Menu:\n");
17         printf("1. Push\n");
18         printf("2. Pop\n");
19         printf("3. Display\n");
20         printf("4. Exit\n");
21         printf("Enter your choice: ");
22         scanf("%d", &choice);
23         switch (choice) {
24             case 1:
25                 printf("Enter value to push: ");
26                 scanf("%d", &value);
27                 push(&stack, value);
28                 break;
29             case 2:
30                 value = pop(&stack);
31                 if (value != -1)
32                     printf("Popped value: %d\n", value);
33                 break;
34             case 3:
35                 display(&stack);
36                 break;
37             case 4:
38                 printf("Exiting...\n");
39                 exit(0);
40             default:
41                 printf("Invalid choice! Please try again.\n");
42         }
43     } while (choice != 4);
44 }
```

BHAVLEEN KAUR

3EIC-2

102155010

```
43     } while (choice != 4);
44     return 0;
45 }
46- void push(struct Stack *s, int value) {
47-     if (s->top == MAX_SIZE - 1) {
48         printf("Stack Overflow! Cannot push element.\n");
49-     } else {
50         s->top++;
51         s->items[s->top] = value;
52         printf("%d pushed onto the stack.\n", value);
53     }
54 }
55- int pop(struct Stack *s) {
56-     if (s->top == -1) {
57         printf("Stack Underflow! Cannot pop element.\n");
58         return -1;
59-     } else {
60         int popped = s->items[s->top];
61         s->top--;
62         return popped;
63     }
64 }
65- void display(struct Stack *s) {
66-     if (s->top == -1) {
67         printf("Stack is empty.\n");
68-     } else {
69         printf("Stack elements: ");
70-         for (int i = s->top; i >= 0; i--) {
71             printf("%d ", s->items[i]);
72         }
73         printf("\n");
74     }
75 }
```

BHAVLEEN KAUR
3EIC-2
102155010

Stack Operations Menu:

1. Push
2. Pop
3. Display
4. Exit

Enter your choice: 1

Enter value to push: 34

34 pushed onto the stack.

Stack Operations Menu:

1. Push
2. Pop
3. Display
4. Exit

Enter your choice: 1

Enter value to push: 23

23 pushed onto the stack.

Stack Operations Menu:

1. Push
2. Pop
3. Display
4. Exit

Enter your choice: 1

Enter value to push: 45

45 pushed onto the stack.

Stack Operations Menu:

1. Push
2. Pop
3. Display
4. Exit

Enter your choice: 2

Popped value: 45

Stack Operations Menu:

1. Push
2. Pop
3. Display
4. Exit

Enter your choice: 2

BHAVLEEN KAUR

3EIC-2

102155010

2. Write a menu driven program with 4 options (Push, Pop, Display, and Exit) to demonstrate the working of stacks using linked-list.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  // Node structure for the linked list
5  struct Node {
6      int data;
7      struct Node* next;
8  };
9
10 // Structure for stack
11 struct Stack {
12     struct Node* top;
13 };
14
15 // Function prototypes
16 void push(struct Stack* s, int value);
17 int pop(struct Stack* s);
18 void display(struct Stack* s);
19
20 int main() {
21     struct Stack stack;
22     stack.top = NULL; // Initialize stack top
23
24     int choice, value;
25
26     do {
27         // Display menu
28         printf("\nStack Operations Menu:\n");
29         printf("1. Push\n");
30         printf("2. Pop\n");
31         printf("3. Display\n");
32         printf("4. Exit\n");
33         printf("Enter your choice: ");
34         scanf("%d", &choice);
35
36         switch (choice) {
37             case 1:
38                 printf("Enter value to push: ");
39                 scanf("%d", &value);
40                 push(&stack, value);
41                 break;
42             case 2:
```

BHAVLEEN KAUR

3EIC-2

102155010

```
43         value = pop(&stack);
44         if (value != -1)
45             printf("Popped value: %d\n", value);
46         break;
47     case 3:
48         display(&stack);
49         break;
50     case 4:
51         printf("Exiting...\n");
52         exit(0);
53     default:
54         printf("Invalid choice! Please try again.\n");
55     }
56 } while (choice != 4);
57
58 return 0;
59 }
60
61 // Function to push an element onto the stack
62 void push(struct Stack* s, int value) {
63     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
64     if (newNode == NULL) {
65         printf("Memory allocation failed. Cannot push element.\n");
66         return;
67     }
68     newNode->data = value;
69     newNode->next = s->top;
70     s->top = newNode;
71     printf("%d pushed onto the stack.\n", value);
72 }
73
74 // Function to pop an element from the stack
75 int pop(struct Stack* s) {
76     if (s->top == NULL) {
77         printf("Stack Underflow! Cannot pop element.\n");
78         return -1;
79     }
80     struct Node* temp = s->top;
81     int popped = temp->data;
82     s->top = temp->next;
83     free(temp);
```

BHAVLEEN KAUR

3EIC-2

102155010

```
84     return popped;
85 }
86
87 // Function to display the elements of the stack
88 void display(struct Stack* s) {
89     if (s->top == NULL) {
90         printf("Stack is empty.\n");
91         return;
92     }
93     printf("Stack elements: ");
94     struct Node* current = s->top;
95     while (current != NULL) {
96         printf("%d ", current->data);
97         current = current->next;
98     }
99     printf("\n");
100 }
```

BHAVLEEN KAUR
3EIC-2
102155010

Stack Operations Menu:

1. Push
2. Pop
3. Display
4. Exit

Enter your choice: 1

Enter value to push: 23

23 pushed onto the stack.

Stack Operations Menu:

1. Push
2. Pop
3. Display
4. Exit

Enter your choice: 1

Enter value to push: 34

34 pushed onto the stack.

Stack Operations Menu:

1. Push
2. Pop
3. Display
4. Exit

Enter your choice: 2

Popped value: 34

Stack Operations Menu:

1. Push
2. Pop
3. Display
4. Exit

Enter your choice: 3

Stack elements: 23

Stack Operations Menu:

1. Push
2. Pop
3. Display
4. Exit

Enter your choice: |

BHAVLEEN KAUR

3EIC-2

102155010

3. Write a program to convert infix expression into postfix expression using stack.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  #define MAX_SIZE 100
6
7  // Structure for stack
8- struct Stack {
9      int top;
10     char items[MAX_SIZE];
11 };
12
13 // Function prototypes
14 void push(struct Stack* s, char value);
15 char pop
16- char pop(struct Stack* s) {
17-     if (s->top == -1) {
18         printf("Stack Underflow! Cannot pop element.\n");
19         exit(EXIT_FAILURE);
20     }
21     return s->items[(s->top)--];
22 }
23
24 // Function to get precedence of operators
25- int precedence(char op) {
26-     switch (op) {
27         case '+':
28         case '-':
29             return 1;
30         case '*':
31         case '/':
32             return 2;
33         default:
34             return 0;
35     }
36 }
37
38 // Function to convert infix expression to postfix expression
39- void infixToPostfix(char* infix, char* postfix) {
40     struct Stack stack;
41     stack.top = -1; // Initialize stack top
```


BHAVLEEN KAUR

3EIC-2

102155010

```
43     int i = 0, j = 0;
44
45     while (infix[i] != '\0') {
46         if (infix[i] >= '0' && infix[i] <= '9') {
47             postfix[j++] = infix[i++];
48         } else if (infix[i] == '(') {
49             push(&stack, infix[i++]);
50         } else if (infix[i] == ')') {
51             while (stack.top != -1 && stack.items[stack.top] != '(') {
52                 postfix[j++] = pop(&stack);
53             }
54             if (stack.top == -1) {
55                 printf("Invalid infix expression. Mismatched parentheses.\n");
56                 exit(EXIT_FAILURE);
57             }
58             pop(&stack); // Discard '(' from stack
59             i++;
60         } else {
61             while (stack.top != -1 && precedence(infix[i]) <= precedence(stack
                .items[stack.top])) {
62                 postfix[j++] = pop(&stack);
63             }
64             push(&stack, infix[i++]);
65         }
66     }
67
68     // Pop remaining operators from the stack
69     while (stack.top != -1) {
70         postfix[j++] = pop(&stack);
71     }
72
73     postfix[j] = '\0'; // Null terminate the postfix expression
74 }
75
```

Enter infix expression: 2 3 4 5 6 7 2 1 3 3

Postfix expression: 23 4 5 6 7 2 1 33

=== Code Execution Successful ===

BHAVLEEN KAUR

3EIC-2

102155010

4. Write a program to convert infix expression into prefix expression using stack.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  #define MAX_SIZE 100
6
7  // Structure for stack
8  struct Stack {
9      int top;
10     char items[MAX_SIZE];
11 };
12
13 // Function prototypes
14 void push(struct Stack* s, char value);
15 char pop(struct Stack* s);
16 int precedence(char op);
17 void infixToPrefix(char* infix, char* prefix);
18 void reverseString(char* str);
19
20 int main() {
21     char infix[MAX_SIZE], prefix[MAX_SIZE];
22
23     // Input infix expression
24     printf("Enter infix expression: ");
25     fgets(infix, MAX_SIZE, stdin);
26
27     // Remove newline character if present
28     if (infix[strlen(infix) - 1] == '\n')
29         infix[strlen(infix) - 1] = '\0';
30
31     infixToPrefix(infix, prefix);
32
33     // Output prefix expression
34     printf("Prefix expression: %s\n", prefix);
35
36     return 0;
37 }
38
39 // Function to push an element onto the stack
40 void push(struct Stack* s, char value) {
41     if (s->top == MAX_SIZE - 1) {
42         printf("Stack Overflow! Cannot push element.\n");
```

BHAVLEEN KAUR

3EIC-2

102155010

```
        printf("Stack Overflow! Cannot push element.\n");
        exit(EXIT_FAILURE);
    }
    s->items[++(s->top)] = value;
}

// Function to pop an element from the stack
char pop(struct Stack* s) {
    if (s->top == -1) {
        printf("Stack Underflow! Cannot pop element.\n");
        exit(EXIT_FAILURE);
    }
    return s->items[(s->top)--];
}

// Function to get precedence of operators
int precedence(char op) {
    switch (op) {
        case '+':
        case '-':
            return 1;
    }
}
```

5. Write a program to evaluate a postfix expression using stack.

BHAVLEEN KAUR

3EIC-2

102155010

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <ctype.h>
4  #define MAX_SIZE 100
5  struct Stack {
6      int top;
7      int items[MAX_SIZE];
8  };
9  void push(struct Stack* s, int value);
10 int pop(struct Stack* s);
11 int evaluatePostfix(char* postfix);
12 int main() {
13     char postfix[MAX_SIZE];
14     printf("Enter postfix expression: ");
15     fgets(postfix, MAX_SIZE, stdin);
16     if (postfix[strlen(postfix) - 1] == '\n')
17         postfix[strlen(postfix) - 1] = '\0';
18     int result = evaluatePostfix(postfix);
19     printf("Result: %d\n", result);
20     return 0;
21 }
22 void push(struct Stack* s, int value) {
23     if (s->top == MAX_SIZE - 1) {
24         printf("Stack Overflow! Cannot push element.\n");
25         exit(EXIT_FAILURE);
26     }
27     s->items[++(s->top)] = value;
28 }
29 int pop(struct Stack* s) {
30     if (s->top == -1) {
31         printf("Stack Underflow! Cannot pop element.\n");
32         exit(EXIT_FAILURE);
33     }
34     return s->items[(s->top)--];
35 }
36 int evaluatePostfix(char* postfix) {
37     struct Stack stack;
38     stack.top = -1;
39     int operand1, operand2, result;
40     for (int i = 0; postfix[i] != '\0'; i++) {
41         if (isdigit(postfix[i])) {
42             push(&stack, postfix[i] - '0');
```

BHAVLEEN KAUR

3EIC-2

102155010

```
43-     } else {
44         operand2 = pop(&stack);
45         operand1 = pop(&stack);
46-         switch (postfix[i]) {
47             case '+':
48                 result = operand1 + operand2;
49                 break;
50             case '-':
51                 result = operand1 - operand2;
52                 break;
53             case '*':
54                 result = operand1 * operand2;
55                 break;
56             case '/':
57-                 if (operand2 == 0) {
58                     printf("Division by zero error.\n");
59                     exit(EXIT_FAILURE);
60                 }
61                 result = operand1 / operand2;
62                 break;
63             default:
64                 printf("Invalid character encountered in postfix expression\n");
65                 exit(EXIT_FAILURE);
66         }
67         push(&stack, result);
68     }
69 }
70 return pop(&stack);
71 }
```