

DATA STRUCTURE – 1

LAB-5

S.Praveen kumar

ch.en.u4aie22048

Creation of circular linked list:

Program:

```
main.c
1 //S.Praveen Kumar
2 //ch.en.u4aie22048
3 //ds-1 Lab-5
4
5 //creation of circular linked list
6 #include<stdio.h>
7 #include<stdlib.h>
8
9 struct Node {
10     int data;
11     struct Node* link;
12 };
13
14 int main() {
15     struct Node* head = NULL;
16     struct Node* temp = NULL;
17     struct Node* current = NULL;
18
19     int num, ele, i;
20     printf("Enter size of list: ");
21     scanf("%d", &num);
22
23     for (i = 0; i < num; i++) {
24         printf("Enter the element: ");
25         scanf("%d", &ele);
26
27         if (head == NULL)
28         {
29             head = (struct Node*) malloc(sizeof(struct Node));
30             head->data = ele;
31             head->link = head;
32             current = head;
33         }
34         else
35         {
36             temp = (struct Node*) malloc(sizeof(struct Node));
37             temp->data = ele;
38             temp->link = head;
39             current->link = temp;
40             current = temp;
41         }
42     }
43
44     printf("List elements: ");
45     current = head;
46     do {
47         printf("%d->", current->data);
48         current = current->link;
49     } while (current != head);
50     printf("NULL");
51     return 0;
52 }
53
```

Output:

```
input
Enter size of list: 5
Enter the element: 1
Enter the element: 2
Enter the element: 3
Enter the element: 4
Enter the element: 5
List elements: 1->2->3->4->5->NULL
...Program finished with exit code 0
Press ENTER to exit console.
```

Deletion of circular linked list:

Program:

```
main.c
1 //S.Praveen Kumar
2 //ch.en.u4a1e22048
3 //ds-1 Lab-5
4
5 //deletion of circular Linked List
6 #include<stdio.h>
7 #include<stdlib.h>
8
9 struct Node {
10     int data;
11     struct Node* link;
12 };
13
14 int main()
15 {
16     struct Node* head = NULL;
17     struct Node* temp = NULL;
18     struct Node* current = NULL;
19
20     int num, ele, i, pos;
21     printf("Enter size of list: ");
22     scanf("%d", &num);
23
24     for (i = 0; i < num; i++)
25     {
26         printf("Enter the element: ");
27         scanf("%d", &ele);
28
29         if (head == NULL)
30         {
31             head = (struct Node*) malloc(sizeof(struct Node));
32             head->data = ele;
33             head->link = head;
34             current = head;
35         } else
36         {
37             temp = (struct Node*) malloc(sizeof(struct Node));
38             temp->data = ele;
39             temp->link = head;
40             current->link = temp;
41             current = temp;
42         }
43     }
44
45     printf("List elements: ");
46     current = head;
47     do {
48         printf("%d->", current->data);
49         current = current->link;
50     } while (current != head);
51     printf("NULL");
52     printf("\n");
53
54     printf("Enter position to delete: ");
55     scanf("%d", &pos);
56
57     if (pos <= 0 || pos > num)
58     {
59         printf("Invalid position.\n");
60     }
61     else if (pos == 1)
62     {
63
64         if (num == 1)
65         {
66             free(head);
67             head = NULL;
68         }
69         else
70         {
71             current = head;
72             while (current->link != head) {
73                 current = current->link;
74             }
75             temp = head;
76             head = head->link;
77             current->link = head;
78             free(temp);
79         }
80     }
81     else
82     {
83         current = head;
84         for (i = 1; i < pos - 1; i++)
85         {
86             current = current->link;
87         }
88         temp = current->link;
89         current->link = temp->link;
90         free(temp);
91     }
92
93     printf("List after deletion: ");
94     if (head == NULL)
95     {
96         printf("List is empty.\n");
97     }
98     else
99     {
100         current = head;
101         do {
102             printf("%d->", current->data);
103             current = current->link;
104         } while (current != head);
105     }
106     printf("NULL");
107     return 0;
108 }
109 }
```

Output:

```
input
Enter size of list: 5
Enter the element: 1
Enter the element: 2
Enter the element: 3
Enter the element: 3
Enter the element: 4
List elements: 1->2->3->3->4->NULL
Enter position to delete:
3
List after deletion: 1->2->3->4->NULL

...Program finished with exit code 0
Press ENTER to exit console.
```

Insertion of circular linked list:

Program:

```
main.c
1 //S.Praveen Kumar
2 //ch.en.u4aie22048
3 //ds-1 Lab-5
4
5 //Insertion of circular Linked List
6 #include<stdio.h>
7 #include<stdlib.h>
8
9 struct Node
10 {
11     int data;
12     struct Node* link;
13 };
14
15 int main()
16 {
17     struct Node* head = NULL;
18     struct Node* temp = NULL;
19     struct Node* current = NULL;
20
21     int num, ele, i, pos;
22     printf("Enter size of list: ");
23     scanf("%d", &num);
24
25     for (i = 0; i < num; i++) {
26         printf("Enter the element: ");
27         scanf("%d", &ele);
28
29         if (head == NULL) {
30             head = (struct Node*) malloc(sizeof(struct Node));
31             head->data = ele;
32             head->link = head;
33             current = head;
34         } else {
35             temp = (struct Node*) malloc(sizeof(struct Node));
36             temp->data = ele;
37             temp->link = head;
38             current->link = temp;
```

```
39         current = temp;
40     }
41 }
42
43 printf("List elements: ");
44 current = head;
45 do {
46     printf("%d->", current->data);
47     current = current->link;
48 } while (current != head);
49 printf("NULL");
50 printf("\n");
51
52 printf("Enter position to insert: ");
53 scanf("%d", &pos);
54 printf("Enter element to insert: ");
55 scanf("%d", &ele);
56
57 if (pos <= 0 || pos > num + 1) {
58     printf("Invalid position.\n");
59 } else {
60     temp = (struct Node*) malloc(sizeof(struct Node));
61     temp->data = ele;
62     if (pos == 1) {
63         if (head == NULL) {
64             head = temp;
65             temp->link = temp;
66         } else {
67             current = head;
68             while (current->link != head) {
69                 current = current->link;
70             }
71             temp->link = head;
72             current->link = temp;
73             head = temp;
74         }
75     } else {
76         current = head;
```

```

77     for (i = 1; i < pos - 1; i++) {
78         current = current->link;
79     }
80     temp->link = current->link;
81     current->link = temp;
82 }
83 num++;
84 }
85
86 printf("List after insertion: ");
87 current = head;
88 do {
89     printf("%d->", current->data);
90     current = current->link;
91 } while (current != head);
92 printf("NULL");
93
94 return 0;
95 }

```

Output:

```

input
Enter size of list: 5
Enter the element: 1
Enter the element: 2
Enter the element: 4
Enter the element: 5
Enter the element: 6
List elements: 1->2->4->5->6->NULL
Enter position to insert: 3
Enter element to insert: 3
List after insertion: 1->2->3->4->5->6->NULL

...Program finished with exit code 0
Press ENTER to exit console.

```

Creation of circular double linked list:

Program:

```

main.c
1 //S.Praveen Kumar
2 //ch.en.u4aie22048
3 //ds-1 Lab-5
4
5 //Creation of circular double Linked List
6 #include <stdio.h>
7 #include <stdlib.h>
8
9 struct Node {
10     int data;
11     struct Node* prev;
12     struct Node* next;
13 };
14
15 int main() {
16     struct Node* head = NULL;
17     struct Node* temp = NULL;
18     struct Node* current = NULL;
19
20     int num, ele, i;
21     printf("Enter size of list: ");
22     scanf("%d", &num);
23
24     for (i = 0; i < num; i++) {
25         printf("Enter the element: ");
26         scanf("%d", &ele);
27
28         if (head == NULL) {
29             head = (struct Node*) malloc(sizeof(struct Node));
30             head->data = ele;
31             head->prev = head;
32             head->next = head;
33             current = head;
34         } else {
35             temp = (struct Node*) malloc(sizeof(struct Node));
36             temp->data = ele;
37             temp->prev = current;
38             temp->next = head;

```

```

39     current->next = temp;
40     head->prev = temp;
41     current = temp;
42 }
43 }
44
45 printf("List elements: ");
46 current = head;
47 do {
48     printf("%d->", current->data);
49     current = current->next;
50 } while (current != head);
51 printf("NULL\n");
52
53 printf("List elements (in reverse): ");
54 printf("NULL");
55 current = head->prev;
56 do {
57     printf("->%d", current->data);
58     current = current->prev;
59 } while (current != head->prev);
60
61 return 0;
62 }

```

Output:

```

input
Enter size of list: 5
Enter the element: 1
Enter the element: 2
Enter the element: 3
Enter the element: 4
Enter the element: 5
List elements: 1->2->3->4->5->NULL
List elements (in reverse): NULL->5->4->3->2->1
...Program finished with exit code 0
Press ENTER to exit console.

```

Insertion of circular double linked list:

Program:

```

main.c
1 //S.Praveen Kumar
2 //ch.en.u4aie22048
3 //ds-1 lab-5
4
5 //Creation of circular double linked list
6 #include<stdio.h>
7 #include<stdlib.h>
8
9 struct Node {
10     int data;
11     struct Node* prev;
12     struct Node* next;
13 };
14
15 int main() {
16     struct Node* head = NULL;
17     struct Node* temp = NULL;
18     struct Node* current = NULL;
19
20     int num, ele, i, pos;
21     printf("Enter size of list: ");
22     scanf("%d", &num);
23
24     for (i = 0; i < num; i++) {
25         printf("Enter the element: ");
26         scanf("%d", &ele);
27
28         if (head == NULL) {
29             head = (struct Node*) malloc(sizeof(struct Node));
30             head->data = ele;
31             head->prev = NULL;
32             head->next = NULL;
33             current = head;
34         } else {
35             temp = (struct Node*) malloc(sizeof(struct Node));
36             temp->data = ele;
37             temp->prev = current;
38             temp->next = NULL;

```

```

39         current->next = temp;
40         current = temp;
41     }
42 }
43
44 printf("List elements: ");
45 current = head;
46 while (current != NULL) {
47     printf("%d->", current->data);
48     current = current->next;
49 }
50 printf("NULL\n");
51
52 printf("Enter position to insert: ");
53 scanf("%d", &pos);
54 printf("Enter element to insert: ");
55 scanf("%d", &ele);
56
57 if (pos <= 0 || pos > num + 1) {
58     printf("Invalid position.\n");
59 } else {
60     temp = (struct Node*) malloc(sizeof(struct Node));
61     temp->data = ele;
62     if (pos == 1) {
63         if (head == NULL) {
64             head = temp;
65             temp->prev = NULL;
66             temp->next = NULL;
67         } else {
68             temp->prev = NULL;
69             temp->next = head;
70             head->prev = temp;
71             head = temp;
72         }
73     } else {
74         current = head;
75         for (i = 1; i < pos - 1; i++) {
76             current = current->next;
77         }
78         temp->prev = current;
79         temp->next = current->next;
80         if (current->next != NULL) {
81             current->next->prev = temp;
82         }
83         current->next = temp;
84     }
85     num++;
86 }
87
88 printf("List after insertion: ");
89 current = head;
90 while (current != NULL) {
91     printf("%d->", current->data);
92     current = current->next;
93 }
94 printf("NULL");
95
96 return 0;
97 }
98

```

Output:

```

input
Enter size of list: 5
Enter the element: 1
Enter the element: 2
Enter the element: 4
Enter the element: 5
Enter the element: 6
List elements: 1->2->4->5->6->NULL
Enter position to insert: 3
Enter element to insert: 3
List after insertion: 1->2->3->4->5->6->NULL

...Program finished with exit code 0
Press ENTER to exit console.

```

Deletion of circular double linked list:

Program:

```
main.c
1 //S.Praveen Kumar
2 //ch.en.u4aie22048
3 //ds-1 lab-5
4
5 //Deletion of circular double linked list
6 #include <stdio.h>
7 #include <stdlib.h>
8
9 struct Node {
10     int data;
11     struct Node* next;
12     struct Node* prev;
13 };
14
15 int main() {
16     struct Node* head = NULL;
17     struct Node* temp = NULL;
18     struct Node* current = NULL;
19
20     int num, ele, i, pos;
21     printf("Enter size of list: ");
22     scanf("%d", &num);
23
24     for (i = 0; i < num; i++) {
25         printf("Enter the element: ");
26         scanf("%d", &ele);
27
28         if (head == NULL) {
29             head = (struct Node*) malloc(sizeof(struct Node));
30             head->data = ele;
31             head->next = head;
32             head->prev = head;
33             current = head;
34         } else {
35             temp = (struct Node*) malloc(sizeof(struct Node));
36             temp->data = ele;
37             temp->next = head;
38             temp->prev = current;
39             current->next = temp;
40             head->prev = temp;
41             current = temp;
42         }
43     }
44
45     printf("List elements: ");
46     current = head;
47     do {
48         printf("%d-", current->data);
49         current = current->next;
50     } while (current != head);
51     printf("NULL\n");
52
53     printf("Enter position to delete: ");
54     scanf("%d", &pos);
55
56     if (pos <= 0 || pos > num) {
57         printf("Invalid position.\n");
58     } else if (pos == 1) {
59
60         if (num == 1) {
61             free(head);
62             head = NULL;
63         } else {
64             current = head;
65             while (current->next != head) {
66                 current = current->next;
67             }
68             temp = head;
69             head = head->next;
70             current->next = head;
71             head->prev = current;
72             free(temp);
73         }
74     } else {
75         current = head;
76         for (i = 1; i < pos; i++) {
```

```

77         current = current->next;
78     }
79     temp = current;
80     current->prev->next = current->next;
81     current->next->prev = current->prev;
82     free(temp);
83 }
84
85 printf("List after deletion: ");
86 if (head == NULL) {
87     printf("List is empty.\n");
88 } else {
89     current = head;
90     do {
91         printf("%d->", current->data);
92         current = current->next;
93     } while (current != head);
94     printf("NULL\n");
95 }
96
97 return 0;
98 }

```

Output:

```

input
Enter size of list: 5
Enter the element: 1
Enter the element: 2
Enter the element: 3
Enter the element: 3
Enter the element: 4
List elements: 1->2->3->3->4->NULL
Enter position to delete: 3
List after deletion: 1->2->3->4->NULL

...Program finished with exit code 0
Press ENTER to exit console.

```


DATA STRUCTURE – 1

LAB-6

S.Praveen kumar

ch.en.u4aie22048

Creation of stack using Array:

Program:

```
main.c
1 //S.Praveen Kumar
2 //ch.en.u4aie22048
3 //ds-1 Lab-6
4
5 //Creating of Stack using Array
6 #include <stdio.h>
7 #include <stdlib.h>
8 int main()
9 {
10     int size;
11     printf("Enter the size of the stack: ");
12     scanf("%d",&size);
13     int stack[size],ele,choice,i,temp,j;
14     while(1)
15     {
16         printf("*****MENU*****\n1.push\n2.pop\n3.display\n4.exit\n");
17         printf("Enter the option: ");
18         scanf("%d",&choice);
19         switch(choice)
20         {
21             case 1:
22             {
23                 if(i<size)
24                 {
25                     printf("enter the element to push: ");
26                     scanf("%d",&ele);
27                     stack[i]=ele;
28                     i++;
29                 }
30                 else
31                 {
32                     printf("Overflow");
33                 }
34                 break;
35             }
36             case 2:
37             {
38                 if(i>=0)
39                 {
40                     i--;
41                     printf("Poped the element\n");
42                 }
43                 else
44                 {
45                     printf("Underflow");
46                 }
47                 break;
48             }
49             case 3:
50             {
51                 for(j=i-1;j>=0;j--)
52                 {
53                     printf("%d ",stack[j]);
54                 }
55                 printf("\n");
56                 break;
57             }
58             case 4:
59             {
60                 return 0;
61             }
62         }
63     }
64     return 0;
65 }
```

Output:

```
input
Enter the size of the stack: 5
*****MENU*****
1.push
2.pop
3.display
4.exit
Enter the option: 1
enter the element to push: 1
*****MENU*****
1.push
2.pop
3.display
4.exit
Enter the option: 1
enter the element to push: 2
*****MENU*****
1.push
2.pop
3.display
4.exit
Enter the option: 1
enter the element to push: 3
*****MENU*****
1.push
2.pop
3.display
4.exit
Enter the option: 1
enter the element to push: 4
*****MENU*****
1.push
2.pop
3.display
4.exit
Enter the option: 2
Poped the element
*****MENU*****
1.push
2.pop
3.display
4.exit
Enter the option: 3
3 2 1
*****MENU*****
```

Creation of Stack using linked list:

Program:

```
main.c
1 //S.Praveen Kumar
2 //ch.en.u4aie22048
3 //ds-1 lab-6
4
5 //Creating of Stack using Linked List
6 #include <stdio.h>
7 #include <stdlib.h>
8
9 struct Node {
10     int data;
11     struct Node* next;
12 };
13
14 struct Node* top = NULL;
15
16 void push(int value);
17 void pop();
18 void display();
19 int size();
20
21 int main() {
22     int choice, value;
23
24     while (1) {
25         printf("\n1. Push\n2. Pop\n3. Display\n4. Size\n5. Exit\n");
26         printf("Enter your choice: ");
27         scanf("%d", &choice);
28
29         switch (choice) {
30             case 1:
31                 printf("Enter the value to be pushed: ");
32                 scanf("%d", &value);
33                 push(value);
34                 break;
35             case 2:
36                 pop();
37                 break;
38             case 3:
```

```

39         display();
40         break;
41     case 4:
42         printf("Size of stack is %d\n", size());
43         break;
44     case 5:
45         exit(0);
46     default:
47         printf("Invalid choice\n");
48     }
49 }
50
51 return 0;
52 }
53
54 void push(int value) {
55     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
56     newNode->data = value;
57     newNode->next = top;
58     top = newNode;
59 }
60
61 void pop() {
62     if (top == NULL) {
63         printf("Stack is empty\n");
64         return;
65     }
66     struct Node* temp = top;
67     top = top->next;
68     free(temp);
69 }
70
71 void display() {
72     if (top == NULL) {
73         printf("Stack is empty\n");
74         return;
75     }
76     struct Node* temp = top;
77     while (temp != NULL) {
78         printf("%d ", temp->data);
79         temp = temp->next;
80     }
81     printf("\n");
82 }
83
84 int size() {
85     int count = 0;
86     struct Node* temp = top;
87     while (temp != NULL) {
88         count++;
89         temp = temp->next;
90     }
91     return count;
92 }
93

```

Output:

```

input
1. Push
2. Pop
3. Display
4. Size
5. Exit
Enter your choice: 1
Enter the value to be pushed: 1

1. Push
2. Pop
3. Display
4. Size
5. Exit
Enter your choice: 1
Enter the value to be pushed: 2

1. Push
2. Pop
3. Display
4. Size
5. Exit
Enter your choice: 1
Enter the value to be pushed: 3

1. Push
2. Pop
3. Display
4. Size
5. Exit
Enter your choice: 1
Enter the value to be pushed: 4

1. Push
2. Pop
3. Display
4. Size
5. Exit
Enter your choice: 2

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

```

```
Enter your choice: 1
Enter the value to be pushed: 3
```

```
1. Push
2. Pop
3. Display
4. Size
5. Exit
```

```
Enter your choice: 1
Enter the value to be pushed: 4
```

```
1. Push
2. Pop
3. Display
4. Size
5. Exit
```

```
Enter your choice: 2
```

```
1. Push
2. Pop
3. Display
4. Size
5. Exit
```

```
Enter your choice: 3
3 2 1
```

```
1. Push
2. Pop
3. Display
4. Size
5. Exit
```

```
Enter your choice: 4
Size of stack is 3
```

```
1. Push
2. Pop
3. Display
4. Size
5. Exit
```

```
Enter your choice: 5
```

```
...Program finished with exit code 0
Press ENTER to exit console.
```

Reversing of Stack using linked list:

Program:

```
main.c
1 //S.Praveen Kumar
2 //ch.en.u4aie22048
3 //ds-1 Lab-6
4
5 //reverse in stack
6 #include <stdio.h>
7 int Push(int size, int* stack, int top)
8 {
9     int n;
10    if (top == size-1)
11    {
12        printf("Stack is Full");
13    }
14    else
15    {
16        top++;
17        printf("Enter element: ");
18        scanf("%d", &n);
19        stack[top] = n;
20    }
21    return top;
22 }
23 int Pop(int* stack, int top)
24 {
25     if (top == -1)
26     {
27         printf("Stack is Empty");
28     }
29     else
30     {
31         printf("Deleted element:%d", stack[top]);
32         top--;
33     }
34     return top;
35 }
36 void Display(int* stack, int top)
37 {
```

```

38     for (int i=top; i>=0; i--)
39     {
40         printf("%d -> ", stack[i]);
41     }
42     printf("NULL");
43 }
44 void Reverse(int* stack, int top)
45 {
46     int i, j, temp;
47     for (i = 0, j = top; i < j; i++, j--)
48     {
49         temp = stack[i];
50         stack[i] = stack[j];
51         stack[j] = temp;
52     }
53     printf("Stack reversed successfully\n");
54 }
55 int main()
56 {
57     int size, choice, num=1, top=-1;
58     printf("Enter the size of stack: ");
59     scanf("%d", &size);
60     int stack[size];
61     while (num)
62     {
63         printf("\nStack Operations\n");
64         printf("1. Push\n");
65         printf("2. Pop\n");
66         printf("3. Display\n");
67         printf("4. Reverse\n");
68         printf("5. Exit\n");
69         printf("Enter choice: ");
70         scanf("%d", &choice);
71         switch(choice)
72         {
73             case 1:
74                 top = Push(size, stack, top);
75                 break;
76             case 2:
77                 top = Pop(stack, top);
78                 break;
79             case 3:
80                 Display(stack, top);
81                 break;
82             case 4:
83                 Reverse(stack, top);
84                 break;
85             case 5:
86                 printf("Exit Successful\n");
87                 num = 0;
88                 break;
89             default:
90                 printf("Invalid choice\n");
91         }
92     }
93     return 0;
94 }

```

Output:

```

Enter the size of stack: 5

Stack Operations
1. Push
2. Pop
3. Display
4. Reverse
5. Exit
Enter choice: 1
Enter element: 1

Stack Operations
1. Push
2. Pop
3. Display
4. Reverse
5. Exit
Enter choice: 1
Enter element: 2

Stack Operations
1. Push
2. Pop
3. Display
4. Reverse
5. Exit
Enter choice: 1
Enter element: 3

Stack Operations
1. Push
2. Pop
3. Display
4. Reverse
5. Exit
Enter choice: 1
Enter element: 4

```

```
Stack Operations
1. Push
2. Pop
3. Display
4. Reverse
5. Exit
Enter choice: 2
Deleted element:4
Stack Operations
1. Push
2. Pop
3. Display
4. Reverse
5. Exit
Enter choice: 3
3 -> 2 -> 1 -> NULL
Stack Operations
1. Push
2. Pop
3. Display
4. Reverse
5. Exit
Enter choice: 4
Stack reversed successfully

Stack Operations
1. Push
2. Pop
3. Display
4. Reverse
5. Exit
Enter choice: 3
1 -> 2 -> 3 -> NULL
Stack Operations
1. Push
2. Pop
3. Display
4. Reverse
5. Exit
Enter choice: 5
Exit Successful

...Program finished with exit code 0
Press ENTER to exit console.
```

DATA STRUCTURE – 1

LAB-7

S.Praveen kumar

ch.en.u4aie22048

Infix to Postfix

Program:

```
main.c
1 //S.Praveen Kumar
2 //ch.en.u4aie22048
3 //ds-1 Lab-6
4
5 //Infix to Postfix
6 #include<stdio.h>
7 #include<ctype.h>
8 char Stack[100];
9 int top = -1;
10 void push(char x)
11 {
12     Stack[++top] = x;
13 }
14 char pop()
15 {
16     if(top == -1)
17     {
18         return -1;
19     }
20     else
21     {
22         return Stack[top--];
23     }
24 }
25 int precedence(char x)
26 {
27     if(x == '(')
28     {
29         return 0;
30     }
31     if (x == '+' || x == '-')
32     {
33         return 1;
34     }
35     if (x == '*' || x == '/')
36     {
37         return 2;
38     }
39 }
40 int main()
41 {
42     char exp[100];
43     char *e;
44     int x;
45     printf("Enter the expression : \n");
46     scanf("%s",exp);
47     e = exp;
48     while (*e != '\0')
49     {
50         if (isalnum(*e))
51         {
52             printf("%c ",*e);
53         }
54         else if (*e == '(')
55         {
56             push(*e);
57         }
58         else if (*e == ')')
59         {
60             while((x = pop()) != '(' )
61             {
62                 printf("%c ",x);
63             }
64         }
65         else
66         {
67             while(precedence(Stack[top]) >= precedence(*e))
68             {
69                 printf("%c ",pop());
70             }
71             push(*e);
72         }
73         e++;
74     }
75     while (top != -1)
76     {
77         printf("%c ",pop());
78     }
79     return 0;
80 }
```

Output:

```
Enter the expression :
5+4*3-2
5 4 3 * + 2 -

...Program finished with exit code 0
Press ENTER to exit console.
```

Expression Evaluation

Program:

```
main.c
1 //S.Praveen Kumar
2 //aie ch.en.udaie22048
3 //ds-1 Lab-7
4
5 //expression evaluation
6 #include<stdio.h>
7 #include<stdlib.h>
8
9 struct Stack{
10     int top;
11     char* stack;
12 };
13 struct Stack_int{
14     int top;
15     int* stack;
16 };
17
18 struct Stack* create(){
19     struct Stack* S = (struct Stack*)malloc(sizeof(struct Stack));
20     S->stack = NULL;
21     S->top = -1;
22     return S;
23 }
24 struct Stack_int* create_int(){
25     struct Stack_int* S = (struct Stack_int*)malloc(sizeof(struct Stack_int));
26     S->stack = NULL;
27     S->top = -1;
28     return S;
29 }
30
31 void display(struct Stack* stack){
32     printf("\n-----stack-----\n");
33     printf("-----\n");
34     for(int i =0;i<stack->top;i++){
35         printf(" %c ->",stack->stack[i]);
36     }
37     printf("\n-----\n");
38 }
39
40 void display_int(struct Stack_int* stack){
41     printf("\n-----stack-int-----\n");
42     printf("-----\n");
43     for(int i =0;i<stack->top;i++){
44         printf(" %d ->",stack->stack[i]);
45     }
46     printf("\n-----\n");
47 }
48
49
50 char top(struct Stack* S){
51     if(S->top==1){return '\0';}
52     return S->stack[0];
53 }
54
55 int top_int(struct Stack_int* S){
56     if(S->top==1){return 0;}
57     return S->stack[0];
58 }
59
60 struct Stack* push(struct Stack* stack, char val){
61     if(stack->top==1){
62         stack->top++;
63         stack->stack = (char*)malloc(sizeof(char)*stack->top+1);
64         stack->stack[0] = val;
65     } else {
66         stack->top++;
67         char* k = (char*)malloc(sizeof(char)*stack->top+1);
68         for(int i = 1;i<stack->top;i++){
69             k[i] = stack->stack[i-1];
70         }
71         k[0] = val;
72         stack->stack = k;
73     }
74     return stack;
75 }
```



```

114     return NULL;
115 }
116 if(S->top == 0){
117     S->top--;
118     S->stack = NULL;
119 } else {
120     S->top--;
121     int* k = (int*)malloc(sizeof(int)*S->top+1);
122     for(int i = 0; i<=S->top; i++){
123         k[i] = S->stack[i+1];
124     }
125     S->stack = k;
126 }
127 return S;
128 }
129
130 struct Stack* reverse(struct Stack* S){
131     struct Stack* rev = create();
132     while(S->top>=0){
133         char val = top(S);
134         rev = push(rev, val);
135         S = pop(S);
136     }
137     free(S);
138     return rev;
139 }
140
141 struct Stack* insert(struct Stack* S, char* exp){
142     int i = 0;
143     while(exp[i] != '\0'){
144         S = push(S, exp[i]);
145         i++;
146     }
147     S = reverse(S);
148     return S;
149 }
150
151 int postfix_eval(struct Stack* S){
152     if(stack->top== -1){
153         stack->top++;
154         stack->stack = (int*)malloc(sizeof(int)*stack->top+1);
155         stack->stack[0] = val;
156     } else {
157         stack->top++;
158         int* k = (int*)malloc(sizeof(int)*stack->top+1);
159         for(int i = 1; i<=stack->top; i++){
160             k[i] = stack->stack[i-1];
161         }
162         k[0] = val;
163         stack->stack = k;
164     }
165     return stack;
166 }
167
168 struct Stack* pop(struct Stack* S){
169     if(S->top == -1){
170         return NULL;
171     }
172     if(S->top == 0){
173         S->top--;
174         S->stack = NULL;
175     } else {
176         S->top--;
177         char* k = (char*)malloc(sizeof(char)*S->top+1);
178         for(int i = 0; i<=S->top; i++){
179             k[i] = S->stack[i+1];
180         }
181         S->stack = k;
182     }
183     return S;
184 }
185
186 struct Stack_int* pop_int(struct Stack_int* S){
187     if(S->top == -1){
188         struct Stack_int* E = create_int();
189         while (S->top!==-1)
190         {
191             if(top(S) == ','){
192                 S = pop(S);
193             }
194             else if(top(S)>='0'&& top(S)<='9'){
195                 int val = 0;
196                 while(top(S)>='0'&& top(S)<='9'&& top(S)!='>',''){
197                     val *= 10;
198                     val += top(S)-48;
199                     S = pop(S);
200                 }
201                 E = push_int(E, val);
202             } else {
203                 char operator = top(S);
204                 S = pop(S);
205                 int a = (int)(top_int(E));
206                 E = pop_int(E);
207                 int b = (int)(top_int(E));
208                 E = pop_int(E);
209                 int val = 0;
210                 printf("%d--%d--|>'\n", a, b, operator);
211                 switch (operator)
212                 {
213                     case '+':
214                         val = a+b;
215                         break;
216                     case '-':
217                         val = b-a;
218                         break;
219                     case '*':
220                         val = b*a;
221                         break;
222                     case '/':
223                         val = b/a;
224                         break;
225                 }
226                 E = push_int(E, val);
227             }
228         }
229     }
230     return E;
231 }

```

```

188         default:
189             break;
190     }
191     E = push_int(E,val);
192 }
193 }
194 return top_int(E);
195 }
196 int main(){
197     struct Stack* S = create();
198     S = insert(S,"48,+ ,7,8,* ,6,-");
199     display(S);
200 }

```

Output:

input

```

-----stack-----
4 -> 8 -> , -> + -> , -> 7 -> , -> 8 -> , -> * -> , -> 6 -> , -> - ->

...Program finished with exit code 0
Press ENTER to exit console.

```

DATA STRUCTURE – 1

LAB-8

S.Praveen kumar

ch.en.u4aie22048

Creation of Queue in array:

Program:

```
main.c
1 //S.Praveen Kumar
2 //ch.en.u4aie22048
3 // data structure Lab-8
4
5 // creation of Queue in array
6 #include<stdio.h>
7 int front=0,last=0,queue[100];
8 void push()
9 {
10     printf("Enter the element insert at beginning: ");
11     scanf("%d",&queue[last]);
12     last++;
13 }
14 int pop()
15 {
16     printf("Successfully popped the element in queue\n");
17     front++;
18     return 0;
19 }
20 int display()
21 {
22     int i;
23     printf(" ");
24     for(i=front;i<last;i++)
25     {
26         printf(" %d\n ",queue[i]);
27     }
28     return 0;
29 }
30 int main()
31 {
32     int choice;
33
34     while(1)
35     {
36         printf("*****MENU*****\n1.push\n2.pop\n3.display\n4.exit\n");
37         printf("Enter the option: ");
38         scanf("%d",&choice);
39         switch(choice)
40         {
41             case 1:
42             {
43                 push();
44                 break;
45             }
46             case 2:
47             {
48                 pop();
49                 break;
50             }
51             case 3:
52             {
53                 display();
54                 break;
55             }
56             case 4:
57             {
58                 return 0;
59             }
60         }
61     }
62     return 0;
63 }
64 }
```

Output:

```
input
*****MENU*****
1.push
2.pop
3.display
4.exit
Enter the option: 1
Enter the element insert at beginning: 1
*****MENU*****
1.push
2.pop
3.display
4.exit
Enter the option: 1
Enter the element insert at beginning: 2
*****MENU*****
1.push
2.pop
3.display
4.exit
Enter the option: 1
Enter the element insert at beginning: 3
*****MENU*****
1.push
2.pop
3.display
4.exit
Enter the option: 1
Enter the element insert at beginning: 4
*****MENU*****
1.push
2.pop
3.display
4.exit
Enter the option: 2
Successfully popped the element in queue
*****MENU*****
*****MENU*****
1.push
2.pop
3.display
4.exit
Enter the option: 3
2
3
4
*****MENU*****
1.push
2.pop
3.display
4.exit
Enter the option:
```

Creation of Queue in Linked List:

Program:

```
main.c
1 //S.Praveen Kumar
2 //ch.en,u4aie22048
3 //ds-1 Lab-8
4
5 //Creation of queue in Linked List
6 #include<stdio.h>
7 #include <stdlib.h>
8 struct Node
9 {
10     int data;
11     struct Node *next;
12 }
13 *front = NULL,*rear = NULL;
14 void insert(int);
15 void delete();
16 void display();
17 void main()
18 {
19     int choice, value;
20     while(1){
21         printf("\n1. Insert\n2. Delete\n3. Display\n4. Exit\n");
22         printf("Enter your choice: ");
23         scanf("%d",&choice);
24         switch(choice)
25         {
26             case 1: printf("Enter the value to be insert: ");
27                     scanf("%d", &value);
28                     insert(value);
29                     break;
30             case 2: delete();
31                     break;
32             case 3: display();
33                     break;
34             default: printf("\nInvalid Input\n");
35         }
36     }
37 }
38 void insert(int value)
```

```

39 - {
40     struct Node *newNode;
41     newNode = (struct Node*)malloc(sizeof(struct Node));
42     newNode->data = value;
43     newNode->next = NULL;
44     if(front == NULL)
45         front = rear = newNode;
46     else
47     {
48         rear->next = newNode;
49         rear = newNode;
50     }
51 }
52 void delete()
53 {
54     if(front == NULL)
55         printf("\nQueue is Empty\n");
56     else
57     {
58         struct Node *temp = front;
59         front = front->next;
60         printf("\nDeleted element: %d\n", temp->data);
61         free(temp);
62     }
63 }
64 void display()
65 {
66     if(front == NULL)
67         printf("\nQueue is Empty!!!\n");
68     else
69     {
70         struct Node *temp = front;
71         while(temp->next != NULL){
72             printf("%d--->", temp->data);
73             temp = temp->next;
74         }
75         printf("%d--->NULL\n", temp->data);
76     }
}

```

Output:

```

1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter the value to be insert: 1

1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter the value to be insert: 2

1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter the value to be insert: 3

1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter the value to be insert: 4

1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 2
Deleted element: 1

1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 3
2--->3--->4--->NULL

```

DATA STRUCTURE – 1

LAB-9

S.Praveen kumar

ch.en.u4aie22048

Creation of Circular Queue in array:

Program:

```
main.c
1 //S.Praveen Kumar
2 //ch.en.u4aie22048
3 //ds-1 lab-9
4
5 //Creation of queue in linked list
6 #include <stdio.h>
7 #include <stdbool.h>
8 #define SIZE 5
9 int items[SIZE];
10 int front = -1, rear = -1;
11 bool isFull()
12 {
13     if ((front == rear + 1) || (front == 0 && rear == SIZE - 1))
14         return true;
15     return false;
16 }
17 bool isEmpty()
18 {
19     if (front == -1)
20         return true;
21     return false;
22 }
23 void enqueue(int element)
24 {
25     if (isFull())
26         printf("\n Queue is full\n");
27     else
28     {
29         if (front == -1)
30             front = 0;
31         rear = (rear + 1) % SIZE;
32         items[rear] = element;
33         printf("\n Inserted -> %d", element);
34     }
35 }
36 int dequeue()
37 {
38     int element;
39     if (isEmpty())
40     {
41         printf("\n Queue is empty\n");
42         return (-1);
43     }
44     else
45     {
46         element = items[front];
47         if (front == rear)
48         {
49             front = -1;
50             rear = -1;
51         }
52         else
53         {
54             front = (front + 1) % SIZE;
55         }
56         printf("\n Deleted element -> %d \n", element);
57         return (element);
58     }
59 }
60 void display()
61 {
62     int i;
63     if (isEmpty())
64         printf("\n Empty Queue\n");
65     else
66     {
67         for (i = front; i != rear; i = (i + 1) % SIZE)
68         {
69             printf("%d ->", items[i]);
70         }
71         printf("%d ", items[i]);
72     }
73 }
74 int main()
75 {
76     int choice, element;
```

```

77     while (1)
78     {
79         printf("\n1.Insert");
80         printf("\n2.Delete");
81         printf("\n3.Display");
82         printf("\n4.Exit");
83         printf("\nEnter your choice: ");
84         scanf("%d", &choice);
85         switch (choice)
86         {
87             case 1:
88                 printf("\nEnter the element to be inserted: ");
89                 scanf("%d", &element);
90                 enqueue(element);
91                 break;
92             case 2:
93                 dequeue();
94                 break;
95             case 3:
96                 display();
97                 break;
98             case 4:
99                 printf("\n Exit\n");
100                return 0;
101            default:
102                printf("\n Invalid input\n");
103        }
104    }
105 }

```

Output:

```

1.Insert
2.Delete
3.Display
4.Exit
Enter your choice: 1

Enter the element to be inserted: 1

    Inserted -> 1
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice: 1

Enter the element to be inserted: 2

    Inserted -> 2
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice: 1

Enter the element to be inserted: 3

    Inserted -> 3
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice: 1

Enter the element to be inserted: 4

    Inserted -> 4
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice: 2

    Deleted element -> 1

1.Insert
2.Delete
3.Display
4.Exit
Enter your choice: 3
2 ->3 ->4
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:

```