Course Name: Data Structures and Algorithms Lab
Course code : MCSE501P
Faculty Name: SARAVANAN R – SCOPE

Lab Assessment – 3

List of programs:

1. Write a program in C using arrays to implement the following: Create two lists A and B to store data (receive the input from key board, A - to store int and B - to store char). Perform the following operations:
    a. Deletion of 1st, last, intermediate elements (get the input from the key board)
    b. Insertion at the beginning, end, intermediate locations (get the input from the key board)
    c. Splitting of list into two equal parts
2. Write a program in C to implement queue and operations on it (enqueue, dequeue, location of front and back pointers, display elements). Get the input from the key board for creating queue.
3. Write a program in C to implement circular queue and operations on it (enqueue, dequeue, location of front and back pointers, display elements). Get the input from the key board for creating queue.
4. Write a program in C to implement stack and operations on the it (push, pop, top element, display elements).
5. Write a program in C to evaluate the postfix form of an algebraic expression using stack. Get the input from the keyboard.
6. Write a program in C to convert infix form of an algebraic expression into postfix form using stack. Get the input from the keyboard.
7. Write a program in C to implement the following data structures: Singly linked list, doubly linked list, singly linked circular list, doubly linked circular list. Perform insertion and deletion on these data structures.
8. Write a program in C to perform radix sort
9. Write a program in C to perform insertion sort
10. Write a program in C to perform merge sort
11. Write a program in C to perform selection sort

1. **Write a program in C using arrays to implement the following: Create two lists A and B to store data (receive the input from key board, A - to store int and B - to store char). Perform the following operations:**

a.Deletion of 1st, last, intermediate elements (get the input from the key board)

b. Insertion at the beginning, end, intermediate locations (get the input from the key board)

c. Splitting of list into two equal parts

Program code:

```
#include <stdio.h>

int actualIndexInt=0;
int actualIndexChar=0;

void menu(){
   printf("\nPress 1. Delete 1st element");
   printf("\nPress 2. Delete last element");
   printf("\nPress 3. Delete element of a given index");
   printf("\nPress 4. Insert element in the begining");
   printf("\nPress 5. Insert element in the end");
   printf("\nPress 6. Insert element at a given index");
   printf("\nPress 7. Split the list");
   printf("\nEnter your choice : ");
}

void printIntArray(int arr[],int n){
   for(int i=0;i<=n;i++)
      if(i!=n)
         printf("%d, ",arr[i]);
      else
         printf("%d ",arr[i]);
   printf("\n");
}

void printCharArray(char arr[],int n){
   for(int i=0;i<=n;i++)
      if(i!=n)
         printf("%c, ",arr[i]);
      else
         printf("%c ",arr[i]);
   printf("\n");
}

int main(){
   int n1;
   int n2;
   int temp;
   char tempc;
   printf("Enter size of integer array : ");
   scanf("%d",&n1);
   actualIndexInt=n1-1;
```

```
int intArr[(n1+5)];
int i;
printf("Enter elements of integer array : ");
for(i=0;i<n1;i++)
   scanf("%d",&intArr[i]);
printf("Enter the size of the character array : ");
scanf("%d",&n2);
actualIndexChar=n2-1;
char charArr[(n2+5)];
printf("Enter elements of character array : ");
for(i=0;i<n2;i++){
   fflush(stdin);
   scanf("%c",&charArr[i]);
}
int loop=1;
while(loop){
   printf("\nPress 1. Operate on integer array");
   printf("\nPress 2. Operate on character list");
   printf("\nPress 3. Exit");
   printf("\nEnter your choice : ");
   int ch1;
   scanf("%d",&ch1);
   switch(ch1){
      case 1: ;
         menu();
         int ch1;
         scanf("%d",&ch1);
         {
         switch(ch1){
            case 1: ;
               temp=intArr[0];
               for(i=1;i<=actualIndexInt;i++)
                  intArr[i-1]=intArr[i];
               printf("\n%d removed successfully from the list ",temp);
               actualIndexInt--;
               printIntArray(intArr,actualIndexInt);
            break;

            case 2: ;
               temp=intArr[actualIndexInt--];
               printf("\n%d removed successfully from the list ",temp);
               printIntArray(intArr,actualIndexInt);
            break;

            case 3: ;
               printf("\nEnter index of the element you want to delete");
               int ind;
               scanf("%d",&ind);
               if(ind<=actualIndexInt){
                  temp=intArr[ind];
                  for(i=ind+1;i<=actualIndexInt;i++)
```

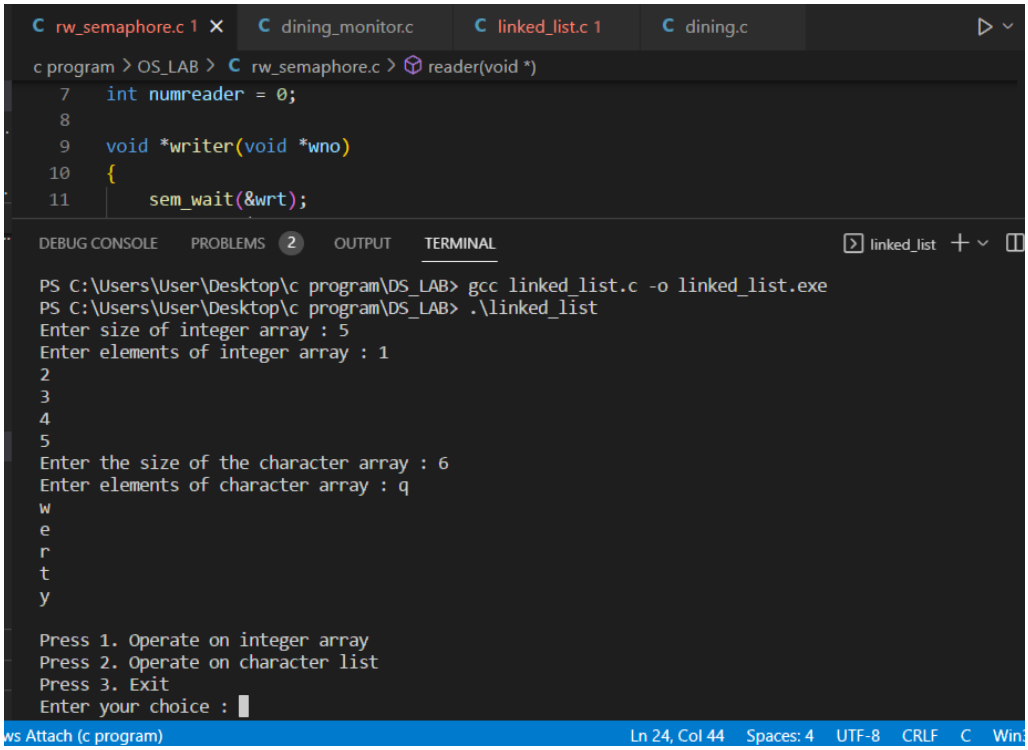
```
            intArr[i-1]=intArr[i];
        printf("\n%d removed successfully from the list ",temp);
        actualIndexInt--;
        printIntArray(intArr,actualIndexInt);
    }
    else
        printf("\nInvalid index");
break;

case 4: ;
    printf("\nEnter element to be inserted ");
    for(i=actualIndexInt;i>=0;i--)
            intArr[i+1]=intArr[i];
    scanf("%d",&intArr[0]);
    actualIndexInt++;
    printf("\nElement addition successful ");
    printIntArray(intArr,actualIndexInt);
break;

case 5: ;
    printf("\nEnter element to be inserted");
    scanf("%d",&intArr[++actualIndexInt]);
    printf("\nElement addition succsful ");
    printIntArray(intArr,actualIndexInt);
break;

case 6: ;
    printf("\nEnter index in which element to be inserted");
    scanf("%d",&temp);
        if(temp<=actualIndexInt){
        for(i=actualIndexInt;i>=temp;i--)
                intArr[i+1]=intArr[i];
        printf("\nEnter element to be inserted");
        scanf("%d",&intArr[temp]);
        actualIndexInt++;
        printf("\nElement added successfully\n");
        printIntArray(intArr,actualIndexInt);
    }
    else
        printf("\nInvalid index");
break;

case 7: ;
    int l1[actualIndexInt/2];
    int l1Len=0;
    int l2[actualIndexInt-(actualIndexInt/2)];
    int l2Len=0;
    for(int i=0;i<=actualIndexInt;i++){
        if(i<=actualIndexInt/2)
            l1[l1Len++]=intArr[i];
        else
```

```
                l2[l2Len++]=intArr[i];
            }
            printf("\nThe splitted arrays are : \n");
            printIntArray(l1,l1Len-1);
            printIntArray(l2,l2Len-1);
        }
          break;

          /*default: ;
            printf("Wrong Input");    */
        }

        break;
    case 2: ;
        menu();
        char ch2;
        fflush(stdin);
        scanf("%c",&ch2);
        {
        switch(ch2){
          case '1': ;
            tempc=charArr[0];
            for(i=1;i<=actualIndexChar;i++)
              charArr[i-1]=charArr[i];
            printf("\n%c removed successfully from the list ",tempc);
            actualIndexChar--;
            printCharArray(charArr,actualIndexChar);
          break;

          case '2': ;
            tempc=charArr[actualIndexChar--];
            printf("\n%c removed successfully from the list ",tempc);
            printCharArray(charArr,actualIndexChar);
          break;

          case '3': ;
            printf("\nEnter index of the element you want to delete ");
            int ind;
            scanf("%d",&ind);
            if(ind<=actualIndexChar){
              tempc=charArr[ind];
              for(i=ind+1;i<=actualIndexChar;i++)
                charArr[i-1]=charArr[i];
              printf("\n%c removed successfully from the list ",tempc);
              actualIndexChar--;
              printCharArray(charArr,actualIndexChar);
            }
            else
              printf("\nInvalid index");
          break;
```

```
        case '4': ;
          printf("\nEnter element to be inserted ");
          for(i=actualIndexChar;i>=0;i--)
                charArr[i+1]=charArr[i];
          fflush(stdin);
          scanf("%c",&charArr[0]);
          actualIndexChar++;
          printf("\nElement addition successful ");
          printCharArray(charArr,actualIndexChar);
        break;

        case '5': ;
          printf("\nEnter element to be inserted ");
          fflush(stdin);
          scanf("%c",&charArr[++actualIndexChar]);
          printf("\nElement addition succsful ");
          printCharArray(charArr,actualIndexChar);
        break;

        case '6': ;
          printf("\nEnter index in which element to be inserted ");

          scanf("%d",&temp);
          if(temp<=actualIndexInt){
            for(i=actualIndexInt;i>=temp;i--)
                  charArr[i+1]=charArr[i];
            printf("\nEnter element to be inserted ");
            fflush(stdin);
            scanf("%c",&charArr[temp]);
            actualIndexChar++;
            printf("\nElement added successfully ");
            printCharArray(charArr,actualIndexChar);
          }
          else
            printf("\nInvalid index");
        break;

        case '7': ;
          char lc1[(actualIndexChar/2)];
          int lc1Len=0;
          char lc2[(actualIndexChar-(actualIndexChar/2))];
          int lc2Len=0;
          for(int i=0;i<=actualIndexChar;i++){
            if(i<=actualIndexChar/2)
              lc1[lc1Len++]=charArr[i];
            else
              lc2[lc2Len++]=charArr[i];
          }
          printf("\nThe splitted arrays are : \n");
          printCharArray(lc1,lc1Len-1);
          printCharArray(lc2,lc2Len-1);
```

```
            break;

            /*/ default: ;
                printf("Wrong Input"); */
        }
        }

        break;
      case 3: ;
        loop = 0;
      break;

      default: ;
        printf("Wrong input");
    }

  }
}
```

Output:- Two lists IntArr and CharArr to store the input from key board, A - to store integer data and B - to store character data.



Operation on Integer list :

1. Deletion of First element :

```
C rw_semaphore.c 1   C dining_monitor.c   C linked_list.c 1 ×   C dining.c
c program > DS_LAB > C linked_list.c > main()
115                 scanf("%d",&intArr[++actualIndexInt]);
116                 printf("\nElement addition succsful ");
117                 printIntArray(intArr,actualIndexInt);
DEBUG CONSOLE   PROBLEMS 2   OUTPUT   TERMINAL                    linked_list

r
t

Press 1. Operate on integer array
Press 2. Operate on character list
Press 3. Exit
Enter your choice : 1

Press 1. Delete 1st element
Press 2. Delete last element
Press 3. Delete element of a given index
Press 4. Insert element in the begining
Press 5. Insert element in the end
Press 6. Insert element at a given index
Press 7. Split the list
Enter your choice : 1

1 removed successfully from the list 2, 3, 4, 5

Press 1. Operate on integer array
Press 2. Operate on character list
Press 3. Exit
```

2. Deletion of last element :

```
C rw_semaphore.c 1   C dining_monitor.c   C linked_list.c 1 ×   C dining.c
c program > DS_LAB > C linked_list.c > main()
115                 scanf("%d",&intArr[++actualIndexInt]);
116                 printf("\nElement addition succsful ");
117                 printIntArray(intArr,actualIndexInt);
DEBUG CONSOLE   PROBLEMS 2   OUTPUT   TERMINAL                    linked_list

1 removed successfully from the list 2, 3, 4, 5

Press 1. Operate on integer array
Press 2. Operate on character list
Press 3. Exit
Enter your choice : 1

Press 1. Delete 1st element
Press 2. Delete last element
Press 3. Delete element of a given index
Press 4. Insert element in the begining
Press 5. Insert element in the end
Press 6. Insert element at a given index
Press 7. Split the list
Enter your choice : 2

5 removed successfully from the list 2, 3, 4

Press 1. Operate on integer array
Press 2. Operate on character list
Press 3. Exit
Enter your choice :
```

3. Delete element of a given index :

```
DEBUG CONSOLE   PROBLEMS 2   OUTPUT   TERMINAL                    linked_list

Press 1. Operate on integer array
Press 2. Operate on character list
Press 3. Exit
Enter your choice : 1

Press 1. Delete 1st element
Press 2. Delete last element
Press 3. Delete element of a given index
Press 4. Insert element in the begining
Press 5. Insert element in the end
Press 6. Insert element at a given index
Press 7. Split the list
Enter your choice : 3

Enter index of the element you want to delete2

4 removed successfully from the list 2, 3

Press 1. Operate on integer array
Press 2. Operate on character list
Press 3. Exit
Enter your choice :
```

4. Insert at beginning

```
C rw_semaphore.c   C dining_monitor.c   C linked_list.c 1   C dining.c
c program > DS_LAB > C linked_list.c > main()
100                 else
101                     printf("\nInvalid index");
102             break;

DEBUG CONSOLE   PROBLEMS 1   OUTPUT   TERMINAL                    powershell

Press 3. Exit
Enter your choice : 1

Press 1. Delete 1st element
Press 2. Delete last element
Press 3. Delete element of a given index
Press 4. Insert element in the begining
Press 5. Insert element in the end
Press 6. Insert element at a given index
Press 7. Split the list
Enter your choice : 4

Enter element to be inserted 8

Element addition successful 8, 1, 2, 3, 4, 5

Press 1. Operate on integer array
Press 2. Operate on character list
Press 3. Exit
Enter your choice :
```

5.Insert at end

```
Press 2. Operate on character list
Press 3. Exit
Enter your choice : 1

Press 1. Delete 1st element
Press 2. Delete last element
Press 3. Delete element of a given index
Press 4. Insert element in the begining
Press 5. Insert element in the end
Press 6. Insert element at a given index
Press 7. Split the list
Enter your choice : 5

Enter element to be inserted9

Element addition succsful 8, 1, 2, 3, 4, 5, 9

Press 1. Operate on integer array
Press 2. Operate on character list
Press 3. Exit
```

6. Insert at given index

```
Press 1. Delete 1st element
Press 2. Delete last element
Press 3. Delete element of a given index
Press 4. Insert element in the begining
Press 5. Insert element in the end
Press 6. Insert element at a given index
Press 7. Split the list
Enter your choice : 6

Enter index in which element to be inserted5

Enter element to be inserted1

Element added successfully
8, 1, 2, 3, 4, 1, 5, 9

Press 1. Operate on integer array
Press 2. Operate on character list
Press 3. Exit
```

7. Split the list

```
DEBUG CONSOLE   PROBLEMS 1   OUTPUT   TERMINAL                powershell

Press 2. Operate on character list
Press 3. Exit
Enter your choice : 1

Press 1. Delete 1st element
Press 2. Delete last element
Press 3. Delete element of a given index
Press 4. Insert element in the begining
Press 5. Insert element in the end
Press 6. Insert element at a given index
Press 7. Split the list
Enter your choice : 7

The splitted arrays are :
8, 1, 2, 3
4, 1, 5, 9

Press 1. Operate on integer array
Press 2. Operate on character list
Press 3. Exit
```

Operation on character list :

1. Deletion of First element :

```
DEBUG CONSOLE   PROBLEMS 1   OUTPUT   TERMINAL                    powershell

Press 1. Operate on integer array
Press 2. Operate on character list
Press 3. Exit
Enter your choice : 2

Press 1. Delete 1st element
Press 2. Delete last element
Press 3. Delete element of a given index
Press 4. Insert element in the begining
Press 5. Insert element in the end
Press 6. Insert element at a given index
Press 7. Split the list
Enter your choice : 1

q removed successfully from the list w, e, r, t

Press 1. Operate on integer array
Press 2. Operate on character list
Press 3. Exit
```

2. Deletion of last element :

```
Press 1. Operate on integer array
Press 2. Operate on character list
Press 3. Exit
Enter your choice : 2

Press 1. Delete 1st element
Press 2. Delete last element
Press 3. Delete element of a given index
Press 4. Insert element in the begining
Press 5. Insert element in the end
Press 6. Insert element at a given index
Press 7. Split the list
Enter your choice : 2

t removed successfully from the list w, e, r
```

3. Delete element of a given index :

```
t removed successfully from the list w, e, r

Press 1. Operate on integer array
Press 2. Operate on character list
Press 3. Exit
Enter your choice : 2

Press 1. Delete 1st element
Press 2. Delete last element
Press 3. Delete element of a given index
Press 4. Insert element in the begining
Press 5. Insert element in the end
Press 6. Insert element at a given index
Press 7. Split the list
Enter your choice : 3

Enter index of the element you want to delete 2

r removed successfully from the list w, e

Press 1. Operate on integer array
Attach (c program)   Ln 112, Col 1   Spaces: 4   UTF-8   CRLF   C   Win32
```

4. Insert at beginning

```
Press 2. Operate on character list
Press 3. Exit
Enter your choice : 2

Press 1. Delete 1st element
Press 2. Delete last element
Press 3. Delete element of a given index
Press 4. Insert element in the begining
Press 5. Insert element in the end
Press 6. Insert element at a given index
Press 7. Split the list
Enter your choice : 4

Enter element to be inserted a

Element addition successful a, w, e

Press 1. Operate on integer array
Press 2. Operate on character list
Press 3. Exit
Enter your choice :
```

5. Insert at end

```
Element addition successful a, w, e

Press 1. Operate on integer array
Press 2. Operate on character list
Press 3. Exit
Enter your choice : 2

Press 1. Delete 1st element
Press 2. Delete last element
Press 3. Delete element of a given index
Press 4. Insert element in the begining
Press 5. Insert element in the end
Press 6. Insert element at a given index
Press 7. Split the list
Enter your choice : 5

Enter element to be inserted f

Element addition succsful a, w, e, f

Press 1. Operate on integer array
```

6. Insert at given index

```
Press 1. Operate on integer array
Press 2. Operate on character list
Press 3. Exit
Enter your choice : 2

Press 1. Delete 1st element
Press 2. Delete last element
Press 3. Delete element of a given index
Press 4. Insert element in the begining
Press 5. Insert element in the end
Press 6. Insert element at a given index
Press 7. Split the list
Enter your choice : 6

Enter index in which element to be inserted 3

Enter element to be inserted j

Element added successfully a, w, e, j, f
```

7. Split the list

```
Element added successfully a, w, e, j, f

Press 1. Operate on integer array
Press 2. Operate on character list
Press 3. Exit
Enter your choice : 2

Press 1. Delete 1st element
Press 2. Delete last element
Press 3. Delete element of a given index
Press 4. Insert element in the begining
Press 5. Insert element in the end
Press 6. Insert element at a given index
Press 7. Split the list
Enter your choice : 7

The splitted arrays are :
a, w, e
j, f
```

2. **Write a program in C to implement queue and operations on it (enqueue, dequeue, location of front and back pointers, display elements). Get the input from the key board for creating queue.**

Code:

```
#include<stdio.h>
void enqueue(int[],int);
int dequeue(int[]);
void display(int[]);

#define SIZE 5
int q[SIZE],front=-1,rear=-1;
void main(){

  int val,ch;
  do{
    printf("\n1.ENQUEUE\n2.DEQUEUE\n3.DISPLAY\n4.EXIT\nENTER YOUR CHOICE:");
    scanf("%d",&ch);
    switch(ch){

      case 1:
        printf("\nEnter value:");
        scanf("%d",&val);
        enqueue(q,val);
      break;

      case 2:
        val=dequeue(q);
        printf("\nDELETED VALUE: %d",val);
      break;

      case 3:
        display(q);
      break;

      default:
```

```
            printf("\nInvalid input");

      }
   }while(ch!=4);
}

void enqueue(int q[], int val){
   if(rear==SIZE-1)
      printf("\nQUEUE IS FULL");
   else if(front==-1){
      front++;
      rear++;
   }
   else
      rear++;

   q[rear]=val;
}

int dequeue(int q[]){
   int val;
   if(front==-1 || front>rear)
      printf("\nQUEUE IS EMPTY");
   val=q[front];
   front++;
   if(front>rear){
      front=-1;
      rear=-1;
   }
   return val;
}

void display(int q[]){
   int i;
   if(front==-1 || front>rear)
      printf("\nQUEUE IS EMPTY");
   else{
      for(i=front;i<=rear;i++)
         printf("%d ",q[i]);
   }
}
```

Output:

1. enqueue

```
DEBUG CONSOLE   PROBLEMS 3   OUTPUT   TERMINAL                queue

1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
ENTER YOUR CHOICE:1

Enter value:1

1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
ENTER YOUR CHOICE:1

Enter value:2

1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
ENTER YOUR CHOICE:3
1 2
1.ENQUEUE
2.DEQUEUE
3.DISPLAY
```

2. dequeue

```
DEBUG CONSOLE   PROBLEMS 3   OUTPUT   TERMINAL                queue

ENTER YOUR CHOICE:3
2 1 2 9
1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
ENTER YOUR CHOICE:2

DELETED VALUE: 2
1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
ENTER YOUR CHOICE:3
1 2 9
1.ENQUEUE
2.DEQUEUE
```

Windows Attach (c program) Ln 56, Col 12 (123 selected) Spaces: 4 UTF-8 CRLF C Win32

3. display elements

```
DEBUG CONSOLE   PROBLEMS 3   OUTPUT   TERMINAL                queue

Enter value:5

QUEUE IS FULL
1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
ENTER YOUR CHOICE:3
1 2 5
1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
ENTER YOUR CHOICE:
```

3. **Write a program in C to implement circular queue and operations on it (enqueue, dequeue, location of front and back pointers, display elements). Get the input from the key board for creating queue.**

Code:-

```
#include<stdio.h>
void enqueue(int[],int);
int dequeue(int[]);
void display(int[]);

#define SIZE 5
int q[SIZE],front=-1,rear=-1;
void main(){
   int val,ch;

   do{
      printf("\n1.ENQUEUE\n2.DEQUEUE\n3.DISPLAY\n4.EXIT\nENTER YOUR CHOICE:");
      scanf("%d",&ch);
      switch(ch){
         case 1:
            printf("\nEnter value:");
            scanf("%d",&val);
            enqueue(q,val);
         break;

         case 2:
         val=dequeue(q);
            printf("\nDELETED VALUE: %d",val);
         break;

         case 3:
            display(q);
         break;

         case 4:
         break;

         default:
            printf("Wrong input");
      }
   }while(ch!=4);
}

void enqueue(int q[], int val){

   if(front==-1 && rear==-1){
         front=rear=0;
         q[rear]=val;
   }
      else if ((rear+1)%SIZE==front ){
```

```
        printf("Queue overflown");
      }
      else{
        rear=(rear+1)%SIZE;
        q[rear]=val;
      }

   }

int dequeue(int q[]){
   int val;
   if(front==-1 || (front==-1 && rear==-1))
      printf("\nQUEUE IS EMPTY");
   else{
      val=q[front];
      if(front==rear)
         front=rear=-1;
      else
         front=(front+1)%SIZE;
      return val;
   }
}
void display(int q[]){
   int i;
   if(front==-1 || (front==-1 && rear==-1))
      printf("\nQUEUE IS EMPTY");
   else{
      for(i=front;i<=rear;i++)
         printf("%d ",q[i]);
   }
}
```

Output:-

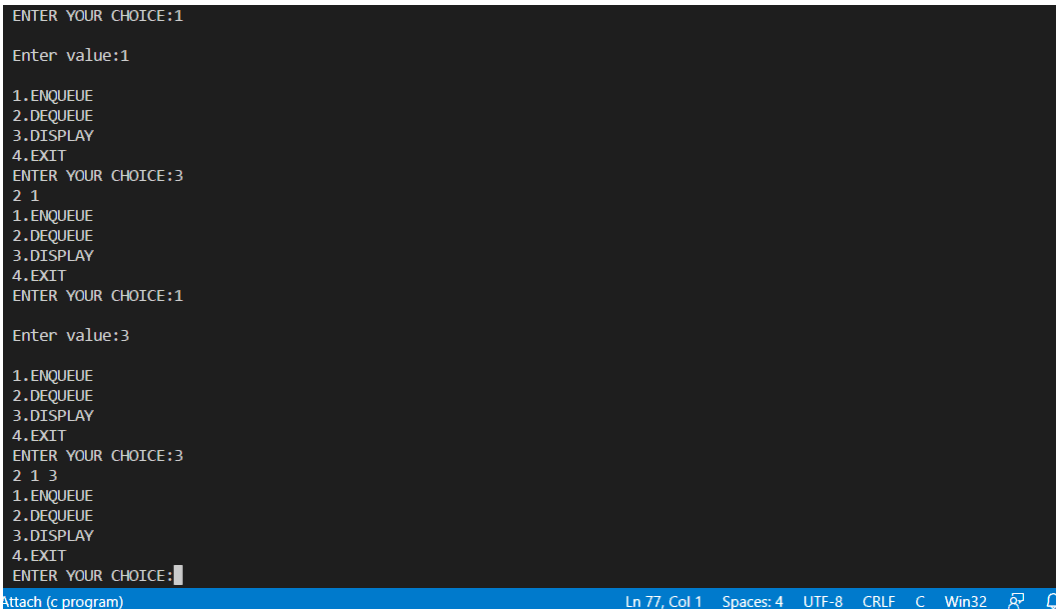
1. enqueue

```
1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
ENTER YOUR CHOICE:1

Enter value:1

1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
ENTER YOUR CHOICE:1

Enter value:2

1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
ENTER YOUR CHOICE:1

Enter value:3

1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
ENTER YOUR CHOICE:3
1 2 3
```

2. **dequeue**

```
4.EXIT
ENTER YOUR CHOICE:3
1 2 3 4 5
1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
ENTER YOUR CHOICE:2

DELETED VALUE: 1
1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
ENTER YOUR CHOICE:2

DELETED VALUE: 2
1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
ENTER YOUR CHOICE:2

DELETED VALUE: 3
1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
ENTER YOUR CHOICE:3
4 5
```

3. Display



```
ENTER YOUR CHOICE:1

Enter value:1

1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
ENTER YOUR CHOICE:3
2 1
1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
ENTER YOUR CHOICE:1

Enter value:3

1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
ENTER YOUR CHOICE:3
2 1 3
1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
ENTER YOUR CHOICE:
```

Attach (c program) Ln 77, Col 1 Spaces: 4 UTF-8 CRLF C Win32

4. Write a program in C to implement stack and operations on the it (push, pop, top element, display elements).

```
#include<stdio.h>
void push(int[],int);
int pop(int[]);
void display(int[]);

#define MAX 5
int s[MAX],top=-1;
void main(){
   int val,ch;
   do{
      printf("\n1.PUSH\n2.POP\n3.DISPLAY\n4.EXIT\nENTER YOUR CHOICE:");
      scanf("%d",&ch);
      switch(ch){
         case 1:
            printf("\nEnter value:");
            scanf("%d",&val);
            push(s,val);
         break;

         case 2:
            val=pop(s);
            printf("\nDELETED VALUE: %d",val);
         break;

         case 3:
```

```
            display(s);
        break;
    }
  }while(ch!=4);
}

void push(int s[], int val){
  if(top==MAX-1)
    printf("\nOVERFLOW");
  else{
    top++;
    s[top]=val;
  }
}

int pop(int s[]){
  int val;
  if(top==-1)
    printf("\nUNDERFLOW");
  else{
    val=s[top];
    top--;
  }
  return val;
}

void display(int s[]){
  int i;
  if(top==-1)
    printf("\nUNDERFLOW");
  else{
    for(i=top;i>=0;i--)
      printf("%d ",s[i]);
  }
}
```

Output:-

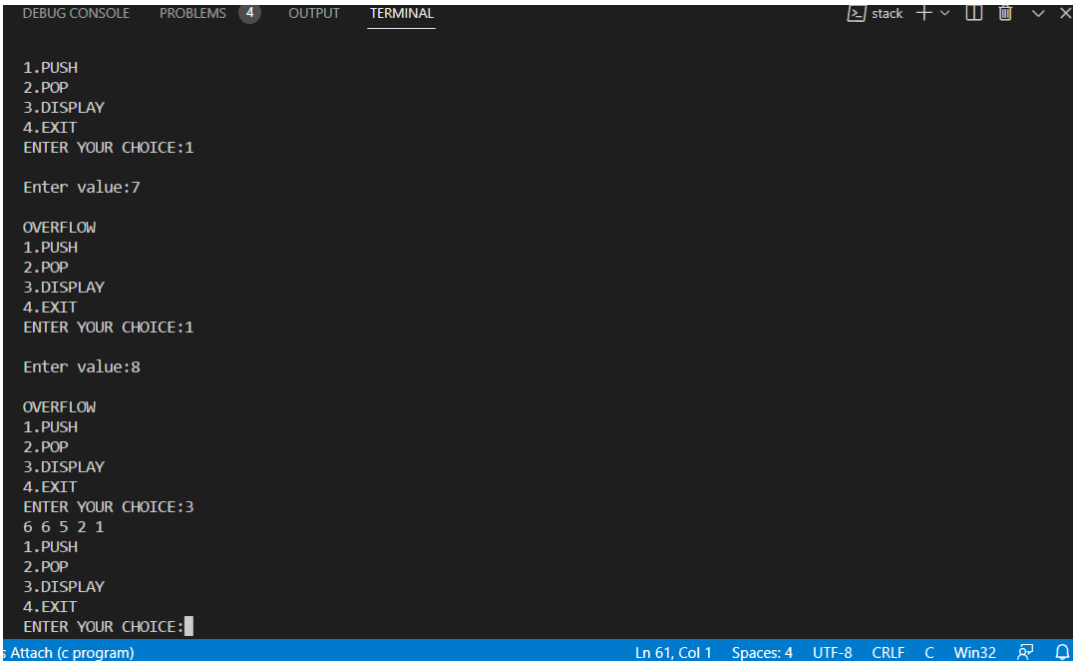
1. push operation

```
ENTER YOUR CHOICE:4
PS C:\Users\User\Desktop\c program\DS_LAB> gcc stack.c -o stack.exe
PS C:\Users\User\Desktop\c program\DS_LAB> .\stack

1.PUSH
2.POP
3.DISPLAY
4.EXIT
ENTER YOUR CHOICE:1

Enter value:1

1.PUSH
2.POP
3.DISPLAY
4.EXIT
ENTER YOUR CHOICE:1

Enter value:2

1.PUSH
2.POP
3.DISPLAY
4.EXIT
ENTER YOUR CHOICE:1

Enter value:3

1.PUSH
2.POP
```

2. Pop Operations

```
2.POP
3.DISPLAY
4.EXIT
ENTER YOUR CHOICE:3
4 3 2 1
1.PUSH
2.POP
3.DISPLAY
4.EXIT
ENTER YOUR CHOICE:2

DELETED VALUE: 4
1.PUSH
2.POP
3.DISPLAY
4.EXIT
ENTER YOUR CHOICE:2

DELETED VALUE: 3
1.PUSH
2.POP
3.DISPLAY
4.EXIT
ENTER YOUR CHOICE:3
2 1
1.PUSH
2.POP
3.DISPLAY
4.EXIT
ENTER YOUR CHOICE:
```

3. Display

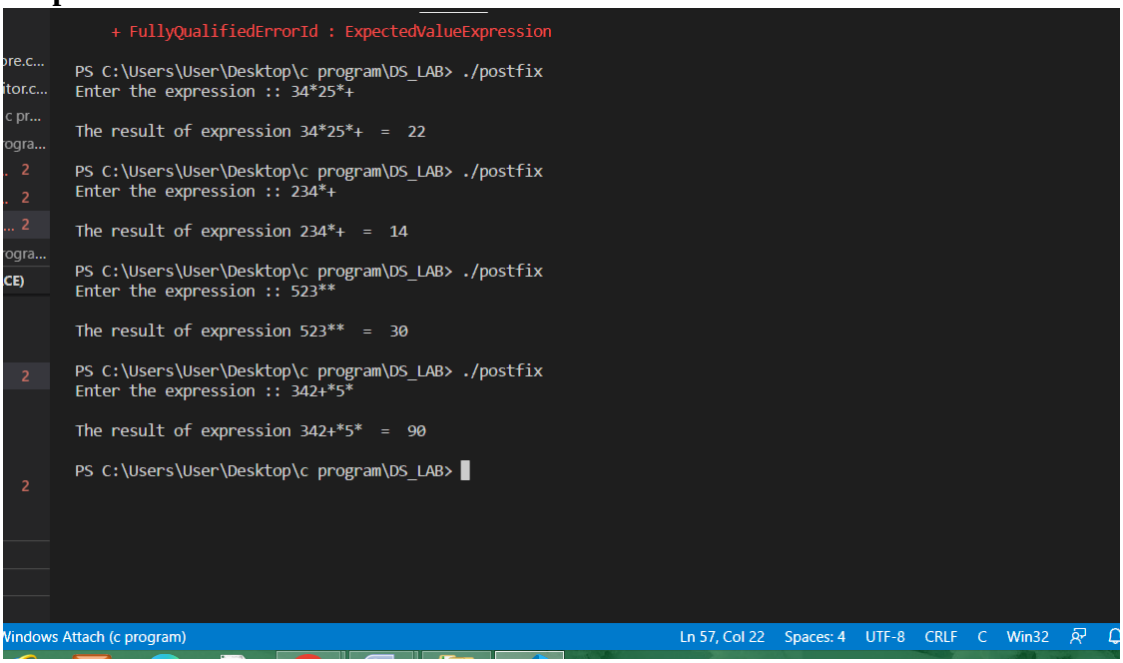


5. Write a program in C to evaluate the postfix form of an algebraic expression using stack. Get the input from the keyboard.

```
 #include<stdio.h>
#include<ctype.h>
int stack[20];
int top = -1;

void push(int x)
{
   stack[++top] = x;
}

int pop()
{
   return stack[top--];
}

int main()
{
   char exp[20];
   char *e;
   int n1,n2,n3,num;
   printf("Enter the expression :: ");
```

```
  scanf("%s",exp);
  e = exp;
  while(*e != '\0')
  {
    if(isdigit(*e))
    {
      num = *e - 48;
      push(num);
    }
    else
    {
      n1 = pop();
      n2 = pop();
      switch(*e)
      {
      case '+':
      {
        n3 = n1 + n2;
        break;
      }
      case '-':
      {
        n3 = n2 - n1;
        break;
      }
      case '*':
      {
        n3 = n1 * n2;
        break;
      }
      case '/':
      {
        n3 = n2 / n1;
        break;
      }
      }
      push(n3);
    }
    e++;
  }
  printf("\nThe result of expression %s  =  %d\n\n",exp,pop());
  return 0;
}
```

Output:-



```
    + FullyQualifiedErrorId : ExpectedValueExpression

PS C:\Users\User\Desktop\c program\DS_LAB> ./postfix
Enter the expression :: 34*25*+

The result of expression 34*25*+  =  22

PS C:\Users\User\Desktop\c program\DS_LAB> ./postfix
Enter the expression :: 234*+

The result of expression 234*+  =  14

PS C:\Users\User\Desktop\c program\DS_LAB> ./postfix
Enter the expression :: 523**

The result of expression 523**  =  30

PS C:\Users\User\Desktop\c program\DS_LAB> ./postfix
Enter the expression :: 342+*5*

The result of expression 342+*5*  =  90

PS C:\Users\User\Desktop\c program\DS_LAB>
```

6. Write a program in C to convert infix form of an algebraic expression into postfix form using stack. Get the input from the keyboard.

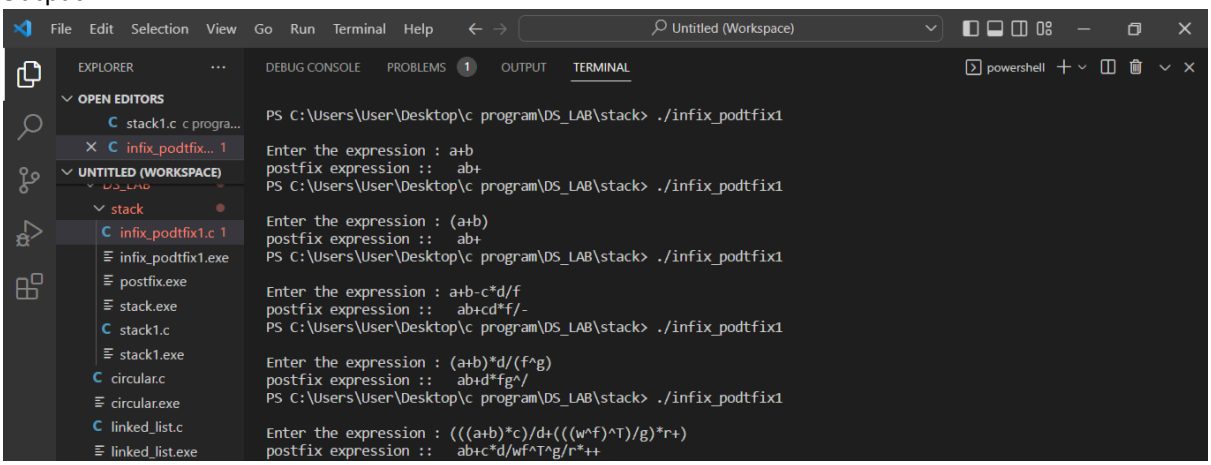
Program:-

```
#include <limits.h>
#include <stdio.h>
#include <stdlib.h>
#define MAX 20
char stk[20];
int top = -1;
int isEmpty()
{
   return top == -1;
}
int isFull()
{
   return top == MAX - 1;
}
char peek()
{
   return stk[top];
}
char pop()
{
   if(isEmpty())
      return -1;
```

```
   char ch = stk[top];
   top--;
   return(ch);
}
void push(char oper)
{
   if(isFull())
      printf("Stack Full!!!!");
   else{
      top++;
      stk[top] = oper;
   }
}
 int checkIfOperand(char ch)
{
   return (ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z');
}
 int precedence(char ch)
{
   switch (ch)
   {
   case '+':
   case '-':
      return 1;
   case '*':
   case '/':
      return 2;
   case '^':
      return 3;
   }
   return -1;
}
 int covertInfixToPostfix(char* expression)
{
   int i, j;
   for (i = 0, j = -1; expression[i]; ++i)
   {
       if (checkIfOperand(expression[i]))
          expression[++j] = expression[i];
       else if (expression[i] == '(')
          push(expression[i]);
       else if (expression[i] == ')')
      {
          while (!isEmpty() && peek() != '(')
             expression[++j] = pop();
          if (!isEmpty() && peek() != '(')
             return -1;
          else
             pop();
      }
      else
```

```
      {
         while (!isEmpty() && precedence(expression[i]) <= precedence(peek()))
            expression[++j] = pop();
         push(expression[i]);
      }
   }
   printf("postfix expression :: \t");
 while (!isEmpty())
      expression[++j] = pop();
   expression[++j] = '\0';
   printf( "%s\n", expression);
}
int main()
{
   char exp[100];
   char *e, x;
   printf("\nEnter the expression : ");
   scanf("%s",exp);
   e = exp;
   covertInfixToPostfix(exp);
   return 0;
}
```

Output:-



7. Write a program in C to implement the following data structures: Singly linked list, doubly linked list, singly linked circular list, doubly linked circular list. Perform insertion and deletion on these data structures.

7.1. Singly linked list

Program:-

```
 #include<stdio.h>
#include<stdlib.h>
struct node
{
   int data;
   struct node *next;
};
struct node *head;
```

```
void beginsert ();
void lastinsert ();
void randominsert();
void begin_delete();
void last_delete();
void random_delete();
void display();
void search();
void main ()
{
   int choice =0;
   while(choice != 9)
   {
      printf("\nChoose one option from the following list ...\n");
       printf("\n1.Insert in begining\n2.Insert at last\n3.Insert at any random location\n4.Delete from
Beginning\n5.Delete from last\n6.Delete node after specified location\n7.Show\n8.Exit\n");
      printf("\nEnter your choice?\n");
      scanf("\n%d",&choice);
      switch(choice)
      {
         case 1:
         beginsert();
         break;
         case 2:
         lastinsert();
         break;
         case 3:
         randominsert();
         break;
         case 4:
         begin_delete();
         break;
         case 5:
         last_delete();
         break;
         case 6:
         random_delete();
         break;
         case 7:
         display();
         break;
         case 8:
         exit(0);
         break;
         default:
         printf("Please enter valid choice..");
      }
   }
}
void beginsert()
```

```
{
   struct node *ptr;
   int item;
   ptr = (struct node *) malloc(sizeof(struct node *));
   if(ptr == NULL)
   {
      printf("\nOVERFLOW");
   }
   else
   {
      printf("\nEnter value\n");
      scanf("%d",&item);
      ptr->data = item;
      ptr->next = head;
      head = ptr;
      printf("\nNode inserted");
   }

}
void lastinsert()
{
   struct node *ptr,*temp;
   int item;
   ptr = (struct node*)malloc(sizeof(struct node));
   if(ptr == NULL)
   {
      printf("\nOVERFLOW");
   }
   else
   {
      printf("\nEnter value?\n");
      scanf("%d",&item);
      ptr->data = item;
      if(head == NULL)
      {
         ptr -> next = NULL;
         head = ptr;
         printf("\nNode inserted");
      }
      else
      {
         temp = head;
         while (temp -> next != NULL)
         {
            temp = temp -> next;
         }
         temp->next = ptr;
         ptr->next = NULL;
         printf("\nNode inserted");

      }
```

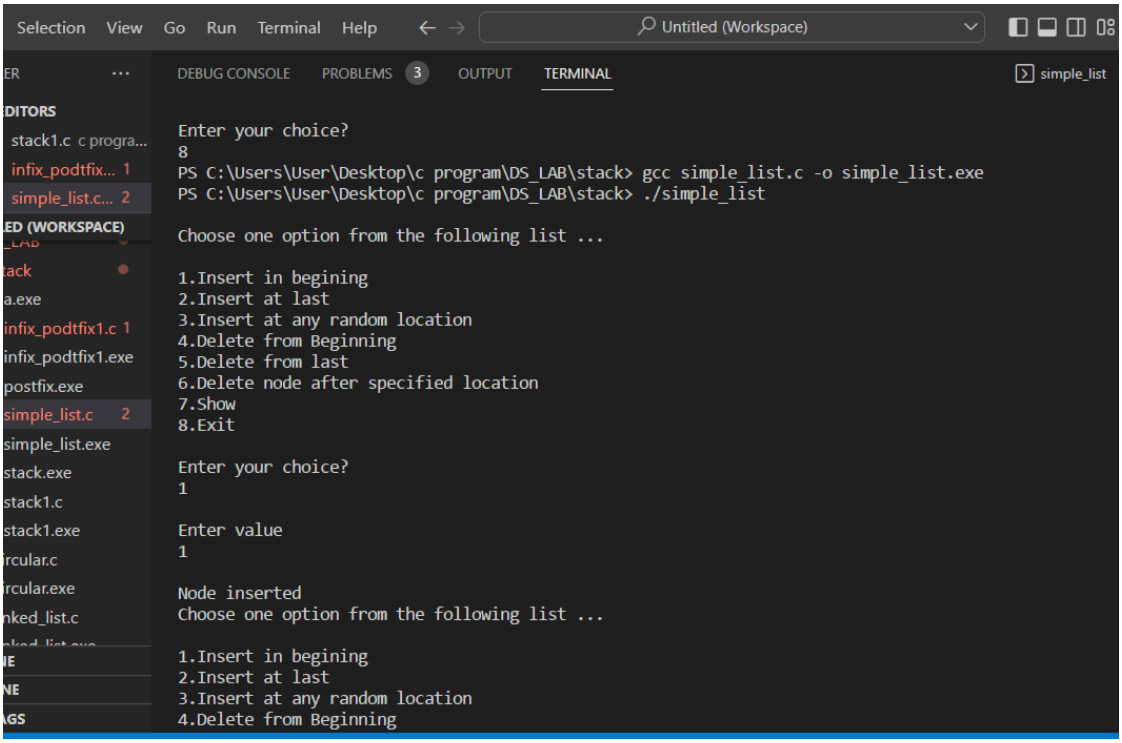
```
    }
}
void randominsert()
{
    int i,loc,item;
    struct node *ptr, *temp;
    ptr = (struct node *) malloc (sizeof(struct node));
    if(ptr == NULL)
    {
        printf("\nOVERFLOW");
    }
    else
    {
        printf("\nEnter element value");
        scanf("%d",&item);
        ptr->data = item;
        printf("\nEnter the location after which you want to insert ");
        scanf("\n%d",&loc);
        temp=head;
        for(i=0;i<loc;i++)
        {
            temp = temp->next;
            if(temp == NULL)
            {
                printf("\ncan't insert\n");
                return;
            }

        }
        ptr ->next = temp ->next;
        temp ->next = ptr;
        printf("\nNode inserted");
    }
}
void begin_delete()
{
    struct node *ptr;
    if(head == NULL)
    {
        printf("\nList is empty\n");
    }
    else
    {
        ptr = head;
        head = ptr->next;
        free(ptr);
        printf("\nNode deleted from the begining ...\n");
    }
}
void last_delete()
{
```

```
   struct node *ptr,*ptr1;
   if(head == NULL)
   {
      printf("\nlist is empty");
   }
   else if(head -> next == NULL)
   {
      head = NULL;
      free(head);
      printf("\nOnly node of the list deleted ...\n");
   }

   else
   {
      ptr = head;
      while(ptr->next != NULL)
      {
         ptr1 = ptr;
         ptr = ptr ->next;
      }
      ptr1->next = NULL;
      free(ptr);
      printf("\nDeleted Node from the last ...\n");
   }
}
void random_delete()
{
   struct node *ptr,*ptr1;
   int loc,i;
   printf("\n Enter the location of the node after which you want to perform deletion \n");
   scanf("%d",&loc);
   ptr=head;
   for(i=0;i<loc;i++)
   {
      ptr1 = ptr;
      ptr = ptr->next;

      if(ptr == NULL)
      {
         printf("\nCan't delete");
         return;
      }
   }
   ptr1 ->next = ptr ->next;
   free(ptr);
   printf("\nDeleted node %d ",loc+1);
}

void display()
{
   struct node *ptr;
```

```
    ptr = head;
    if(ptr == NULL)
    {
        printf("Nothing to print");
    }
    else
    {
        printf("\nprinting values . . . . .\n");
        while (ptr!=NULL)
        {
            printf("\t%d",ptr->data);
            ptr = ptr -> next;
        }
    }
}
```

Output:-

Insertion at beginning:-



Insertion at last:

```
4.Delete from Beginning
5.Delete from last
6.Delete node after specified location
7.Show
8.Exit

Enter your choice?
2

Enter value?
3

Node inserted
Choose one option from the following list ...

1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning
5.Delete from last
6.Delete node after specified location
7.Show
8.Exit

Enter your choice?
7

printing values . . . . .
        8       6       5       1       2       3
Choose one option from the following list ...
```

Insertion at any location:-

```
Enter your choice?
3

Enter element value3

Enter the location after which you want to insert 1

Node inserted
Choose one option from the following list ...

1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning
5.Delete from last
6.Delete node after specified location
7.Show
8.Exit

Enter your choice?
7

printing values . . . . .
        1       2       3
Choose one option from the following list ...
```

Deletion:-

List before deletion:-

```
Enter your choice?
7

printing values . . . . .
        4       8       6       5       3       1       2       3
Choose one option from the following list ...
```

Deletion at beginning:

```
6.Delete node after specified location
7.Show
8.Exit

Enter your choice?
4

Node deleted from the begining ...

Choose one option from the following list ...

1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning
5.Delete from last
6.Delete node after specified location
7.Show
8.Exit

Enter your choice?
7

printing values . . . . .
        8       6       5       3       1       2       3
Choose one option from the following list ...

1.Insert in begining
2.Insert at last
```

Deletion at last

```
4.Delete from Beginning
5.Delete from last
6.Delete node after specified location
7.Show
8.Exit

Enter your choice?
5

Deleted Node from the last ...

Choose one option from the following list ...

1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning
5.Delete from last
6.Delete node after specified location
7.Show
8.Exit

Enter your choice?
7

printing values . . . . .
        8       6       5       3       1       2
Choose one option from the following list ...

1.Insert in begining
```

Deletion at given location:-

```
Go  Run  Terminal  Help   Untitled (Workspace)
DEBUG CONSOLE   PROBLEMS 3   OUTPUT   TERMINAL   simple_list
7.Show
8.Exit

Enter your choice?
6

 Enter the location of the node after which you want to perform deletion
3

Deleted node 4
Choose one option from the following list ...

1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning
5.Delete from last
6.Delete node after specified location
7.Show
8.Exit

Enter your choice?
7

printing values . . . . .
        8       6       5       1       2
Choose one option from the following list ...

1.Insert in begining
2.Insert at last
```

7.2. doubly linked list
Code:-

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
struct node* prev;
int data;
struct node* next;
};
struct node* head = NULL;
void insert_at_beginning(int);
void insert_at_end(int);
void insert_at_position(int, int);
void delete_from_beginning();
void delete_from_position(int);
void delete_from_end();
void print_from_beginning();
void print_from_end(struct node*);
void search_data(int);
void update_node_data(int, int);
void list_sort();
struct node* create_node(int);
int size_of_list();
int getData();
int getPosition();
void empty_list_message();
void memory_error_message();
void invalid_position_message();
int main()
{
// char user_active = 'Y';
int user_choice;
int data, position;
while (1)
{
printf("---------------Nidhi singh 22MAI0015------------------");
printf("\n\n------ Doubly Linked List -------\n");
printf("\n1. Insert a node at the beginning");
printf("\n2. Insert a node at the end");
printf("\n3. Insert a node at the given position");
printf("\n\n4. Delete a node from the beginning");
printf("\n5. Delete a node from the end");
printf("\n6. Delete a node from the given position");
printf("\n\n7. Print list from the beginning");
printf("\n8. Search a node data");
printf("\n9. Exit");
```

```
printf("\n\n-----------------------------\n");
printf("\nEnter your choice: ");
scanf("%d", &user_choice);
printf("\n-----------------------------\n");
switch(user_choice)
{
case 1:
printf("\nInserting a node at beginning");
data = getData();
insert_at_beginning(data);
break;
case 2:
printf("\nInserting a node at end");
data = getData();
insert_at_end(data);
break;
case 3:
printf("\nInserting a node at the given position");
data = getData();
position = getPosition();
insert_at_position(data, position);
break;
case 4:
printf("\nDeleting a node from beginning\n");
delete_from_beginning();
break;
case 5:
printf("\nDeleting a node from end\n");
delete_from_end();
break;
case 6:
printf("\nDelete a node from given position\n");
position = getPosition();
delete_from_position(position);
break;
case 7:
printf("\nPrinting the list from beginning\n\n");
print_from_beginning();
break;
case 8:
printf("\nSearching the node data");
data = getData();
search_data(data);
break;
case 9:
printf("\nProgram was terminated\n\n");
return 0;
default:
printf("\n\tInvalid Choice\n");
}
printf("\n.............................\n");
```

```
// printf("\nDo you want to continue? (Y/N) : ");
// fflush(stdin);
// scanf(" %c", &user_active);
}
return 0;
}
void memory_error_message()
{
printf("\nMemory was not allocated!\n");
}
void invalid_position_message()
{
printf("\nInvalid position!\n");
}
void empty_list_message()
{
printf("\nList is Empty!\n");
}
struct node* create_node(int data)
{
struct node* new_node = (struct node*) malloc(sizeof(struct
node));
if (new_node == NULL)
{
return NULL;
}
else
{
new_node->prev = NULL;
new_node->data = data;
new_node->next = NULL;
}
}
void insert_at_beginning(int data)
{
struct node* new_node = create_node(data);
if (new_node == NULL)
{
memory_error_message();
return;
}
else if(head == NULL)
{
head = new_node;
}
else
{
new_node->next = head;
head->prev = new_node;
head = new_node;
}
```

```
printf("\n* Node with data %d was inserted \n", data);
}
void insert_at_end(int data)
{
struct node* new_node = create_node(data);
if (new_node == NULL)
{
memory_error_message();
return;
}
else if (head == NULL)
{
head = new_node;
}
else
{
struct node* temp = head;
while (temp->next != NULL)
{
temp = temp = temp->next;
}
temp->next = new_node;
new_node->prev = temp;
}
printf("\n* Node with data %d was inserted \n", data);
}
void insert_at_position(int data, int pos)
{
struct node* new_node = create_node(data);
int size = size_of_list();
if (new_node == NULL)
{
memory_error_message();
return;
}
else if (head != NULL && (pos < 1 || pos > size))
{
invalid_position_message();
return;
}
else if (head == NULL && pos == 1)
{
head = new_node;
}
else if (head != NULL && pos == 1)
{
new_node->next = head;
head->prev = new_node;
head = new_node;
}
else
```

```
{
struct node* temp = head;
int count = 1;
while (++count < pos)
{
temp = temp->next;
}
temp->next->prev = new_node;
new_node->next = temp->next;
temp->next = new_node;
new_node->prev = temp;
}
printf("\n* Node with data %d was inserted \n", data);
}
void delete_from_beginning()
{
if (head == NULL)
{
empty_list_message();
return;
}
struct node* temp = head;
head = head->next;
int data = temp->data;
free(temp);
printf("\n* Node with data %d was deleted \n", data);
}
void delete_from_end()
{
if (head == NULL)
{
empty_list_message();
return;
}
struct node* temp = head;
int data = 0;
while (temp->next != NULL)
{
temp = temp->next;
}
if (temp->prev == NULL)
{
head = NULL;
}
else
{
temp->prev->next = temp->next;
}
data = temp->data;
free(temp);
printf("\n* Node with data %d was deleted \n", data);
```

```
}
void delete_from_position(int pos)
{
if (head == NULL)
{
empty_list_message();
return;
}
int size = size_of_list();
struct node* temp = head;
int data = 0;
if (pos < 1 || pos > size)
{
invalid_position_message();
return;
}
else if (pos == 1)
{
head = head->next;
data = head->data;
free(temp);
printf("\n* Node with data %d was deleted \n", data);
}
else
{
int count = 0;
while (++count < pos)
{
temp = temp->next;
}
temp->prev->next = temp->next;
if (pos != size)
{
temp->next->prev = temp->prev;
}
data = temp->data;
free(temp);
printf("\n* Node with data %d was deleted \n", data);
}
}
void print_from_beginning()
{
struct node* temp = head;
while (temp != NULL)
{
printf("%d ", temp->data);
temp = temp->next;
}
}
void search_data(int data)
{
```

```
struct node* temp = head;
int position = 0;
int flag = 0;
while (temp != NULL)
{
position += 1;
if (temp->data == data)
{
flag = 1;
break;
}
temp = temp->next;
}
if (flag == 0)
{
printf("\nNode with data %d was not found\n", data);
}
else
{
printf("\nNode found at %d position\n", position);
}
}
int getData()
{
int data;
printf("\n\nEnter Data: ");
scanf("%d", &data);
return data;
}
int getPosition()
{
int position;
printf("\nEnter Position: ");
scanf("%d", &position);
return position;
}
int size_of_list()
{
struct node* temp = head;
int count = 0;
while (temp != NULL)
{
count += 1;
temp = temp->next;
}
return count;
}
```

Output:-

Insertion at beginning:-

```
DEBUG CONSOLE   PROBLEMS 6   OUTPUT   TERMINAL                d_link
8. Search a node data
9. Exit
----------------------------
Enter your choice: 1

----------------------------
Inserting a node at beginning

Enter Data: 1

* Node with data 1 was inserted

...............................--------------Nidhi singh 22MAI0015-----------------
------ Doubly Linked List -------
1. Insert a node at the beginning
2. Insert a node at the end
3. Insert a node at the given position
4. Delete a node from the beginning
5. Delete a node from the end
6. Delete a node from the given position
7. Print list from the beginning
8. Search a node data
9. Exit
----------------------------
Enter your choice: 7

----------------------------
Printing the list from beginning        1
...............................--------------Nidhi singh 22MAI0015-----------------
------ Doubly Linked List -------
```

Insertion at last:-

```
8. Search a node data
9. Exit
----------------------------
Enter your choice: 2

----------------------------
Inserting a node at end

Enter Data: 2

* Node with data 2 was inserted

...............................--------------Nidhi singh 22MAI0015-----------------
------ Doubly Linked List -------
1. Insert a node at the beginning
2. Insert a node at the end
3. Insert a node at the given position
4. Delete a node from the beginning
5. Delete a node from the end
6. Delete a node from the given position
7. Print list from the beginning
8. Search a node data
9. Exit
----------------------------
Enter your choice: 7

----------------------------
Printing the list from beginning        1 2
...............................--------------Nidhi singh 22MAI0015-----------------
```

Insertion at given location:-

```
Enter your choice: 3

----------------------------
Inserting a node at the given position

Enter Data: 3

Enter Position: 1

* Node with data 3 was inserted

...............................--------------Nidhi singh 22MAI0015-----------------
------ Doubly Linked List -------
1. Insert a node at the beginning
2. Insert a node at the end
3. Insert a node at the given position
4. Delete a node from the beginning
5. Delete a node from the end
6. Delete a node from the given position
7. Print list from the beginning
8. Search a node data
9. Exit
----------------------------
Enter your choice: 7

----------------------------
Printing the list from beginning        3 1 2
...............................--------------Nidhi singh 22MAI0015-----------------
------ Doubly Linked List -------
1. Insert a node at the beginning
```

Deletion at beginning

```
DEBUG CONSOLE   PROBLEMS 6   OUTPUT   TERMINAL                d_link
6. Delete a node from the given position
7. Print list from the beginning
8. Search a node data
9. Exit
----------------------------
Enter your choice: 4

----------------------------
Deleting a node from beginning
* Node with data 3 was deleted

...............................--------------Nidhi singh 22MAI0015-----------------
------ Doubly Linked List -------
1. Insert a node at the beginning
2. Insert a node at the end
3. Insert a node at the given position
4. Delete a node from the beginning
5. Delete a node from the end
6. Delete a node from the given position
7. Print list from the beginning
8. Search a node data
9. Exit
----------------------------
Enter your choice: 7

----------------------------
Printing the list from beginning        1 2
...............................--------------Nidhi singh 22MAI0015-----------------
------ Doubly Linked List -------
1. Insert a node at the beginning
```

Deletion at last

```
4. Delete a node from the beginning
5. Delete a node from the end
6. Delete a node from the given position
7. Print list from the beginning
8. Search a node data
9. Exit
----------------------------
Enter your choice: 5

----------------------------
Deleting a node from end

* Node with data 2 was deleted
...............................--------------Nidhi singh 22MAI0015-----------------
------ Doubly Linked List -------
1. Insert a node at the beginning
2. Insert a node at the end
3. Insert a node at the given position
4. Delete a node from the beginning
5. Delete a node from the end
6. Delete a node from the given position
7. Print list from the beginning
8. Search a node data
9. Exit
----------------------------
Enter your choice: 7

----------------------------
Printing the list from beginning        1
...............................--------------Nidhi singh 22MAI0015-----------------
```

Deletion at given location

```
1. Insert a node at the beginning
2. Insert a node at the end
3. Insert a node at the given position
4. Delete a node from the beginning
5. Delete a node from the end
6. Delete a node from the given position
7. Print list from the beginning
8. Search a node data
9. Exit
----------------------------
Enter your choice: 6

----------------------------
Delete a node from given position
Enter Position: 2

Invalid position!
...............................--------------Nidhi singh 22MAI0015-----------------
------ Doubly Linked List -------
1. Insert a node at the beginning
2. Insert a node at the end
3. Insert a node at the given position
4. Delete a node from the beginning
5. Delete a node from the end
6. Delete a node from the given position
7. Print list from the beginning
8. Search a node data
9. Exit
----------------------------
Enter your choice:
```

Output:-

Insertion at beginning:-

```
DEBUG CONSOLE   PROBLEMS 6   OUTPUT   TERMINAL                d_link

8. Search a node data
9. Exit
----------------------------
Enter your choice: 1

----------------------------
Inserting a node at beginning

Enter Data: 1

* Node with data 1 was inserted

.............................--------------Nidhi singh 22MAI0015-----------------
------ Doubly Linked List -------
1. Insert a node at the beginning
2. Insert a node at the end
3. Insert a node at the given position
4. Delete a node from the beginning
5. Delete a node from the end
6. Delete a node from the given position
7. Print list from the beginning
8. Search a node data
9. Exit
----------------------------
Enter your choice: 7

----------------------------
Printing the list from beginning        1
.............................--------------Nidhi singh 22MAI0015-----------------
------ Doubly Linked List -------
```

Insertion at last:-

```
8. Search a node data
9. Exit
----------------------------
Enter your choice: 2

----------------------------
Inserting a node at end

Enter Data: 2

* Node with data 2 was inserted

.............................--------------Nidhi singh 22MAI0015-----------------
------ Doubly Linked List -------
1. Insert a node at the beginning
2. Insert a node at the end
3. Insert a node at the given position
4. Delete a node from the beginning
5. Delete a node from the end
6. Delete a node from the given position
7. Print list from the beginning
8. Search a node data
9. Exit
----------------------------
Enter your choice: 7

----------------------------
Printing the list from beginning        1 2
.............................--------------Nidhi singh 22MAI0015-----------------
```

Insertion at given location:-

```
Enter your choice: 3

----------------------------
Inserting a node at the given position

Enter Data: 3

Enter Position: 1

* Node with data 3 was inserted

.............................--------------Nidhi singh 22MAI0015-----------------
------ Doubly Linked List -------
1. Insert a node at the beginning
2. Insert a node at the end
3. Insert a node at the given position
4. Delete a node from the beginning
5. Delete a node from the end
6. Delete a node from the given position
7. Print list from the beginning
8. Search a node data
9. Exit
----------------------------
Enter your choice: 7

----------------------------
Printing the list from beginning        3 1 2
.............................--------------Nidhi singh 22MAI0015-----------------
------ Doubly Linked List -------
1. Insert a node at the beginning
```

Deletion at beginning

```
DEBUG CONSOLE   PROBLEMS 6   OUTPUT   TERMINAL                d_link
6. Delete a node from the given position
7. Print list from the beginning
8. Search a node data
9. Exit
----------------------------
Enter your choice: 4

----------------------------
Deleting a node from beginning
* Node with data 3 was deleted

...............................--------------Nidhi singh 22MAI0015-----------------
------ Doubly Linked List -------
1. Insert a node at the beginning
2. Insert a node at the end
3. Insert a node at the given position
4. Delete a node from the beginning
5. Delete a node from the end
6. Delete a node from the given position
7. Print list from the beginning
8. Search a node data
9. Exit
----------------------------
Enter your choice: 7

----------------------------
Printing the list from beginning        1 2
...............................--------------Nidhi singh 22MAI0015-----------------
------ Doubly Linked List -------
1. Insert a node at the beginning
```

Deletion at last

```
4. Delete a node from the beginning
5. Delete a node from the end
6. Delete a node from the given position
7. Print list from the beginning
8. Search a node data
9. Exit
----------------------------
Enter your choice: 5

----------------------------
Deleting a node from end

* Node with data 2 was deleted
...............................--------------Nidhi singh 22MAI0015-----------------
------ Doubly Linked List -------
1. Insert a node at the beginning
2. Insert a node at the end
3. Insert a node at the given position
4. Delete a node from the beginning
5. Delete a node from the end
6. Delete a node from the given position
7. Print list from the beginning
8. Search a node data
9. Exit
----------------------------
Enter your choice: 7

----------------------------
Printing the list from beginning        1
...............................--------------Nidhi singh 22MAI0015-----------------
```

Deletion at given location

```
1. Insert a node at the beginning
2. Insert a node at the end
3. Insert a node at the given position
4. Delete a node from the beginning
5. Delete a node from the end
6. Delete a node from the given position
7. Print list from the beginning
8. Search a node data
9. Exit
----------------------------
Enter your choice: 6

----------------------------
Delete a node from given position
Enter Position: 2

Invalid position!
...............................--------------Nidhi singh 22MAI0015-----------------
------ Doubly Linked List -------
1. Insert a node at the beginning
2. Insert a node at the end
3. Insert a node at the given position
4. Delete a node from the beginning
5. Delete a node from the end
6. Delete a node from the given position
7. Print list from the beginning
8. Search a node data
9. Exit
----------------------------
Enter your choice:
```

7.3. singly linked circular list

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
int data;
struct node *next;
};
struct node *head;
void beginsert ();
void lastinsert ();
void randominsert();
void begin_delete();
void last_delete();
void random_delete();
void display();
void search();
void main ()
{
int choice =0;
while(choice != 7)
{
printf("\n*********Main Menu*********\n");
printf("\nChoose one option from the following list ...\n");
printf("\n==============================================\n");
printf("\n1.Insert in begining\n2.Insert at last\n3.Delete from Beginning\n");
printf("4.Delete from last\n5.Search for an element\n6.Show\n7.Exit\n");
printf("\nEnter your choice?\n");
scanf("\n%d",&choice);
switch(choice)
{
case 1:
beginsert();
break;
case 2:
lastinsert();
break;
case 3:
begin_delete();
break;
case 4:
last_delete();
break;
case 5:
search();
break;
```

```
case 6:
display();
break;
case 7:
exit(0);
break;
default:
printf("Please enter valid choice..");
}
}
}
void beginsert()
{
struct node *ptr,*temp;
int item;
ptr = (struct node *)malloc(sizeof(struct node));
if(ptr == NULL)
{
printf("\nOVERFLOW");
}
else
{
printf("\nEnter the node data?");
scanf("%d",&item);
ptr -> data = item;
if(head == NULL)
{
head = ptr;
ptr -> next = head;
}
else
{
temp = head;
while(temp->next != head)
temp = temp->next;
ptr->next = head;
temp -> next = ptr;
head = ptr;
}
printf("\nnode inserted\n");
}
}
void lastinsert()
{
struct node *ptr,*temp;
int item;
ptr = (struct node *)malloc(sizeof(struct node));
if(ptr == NULL)
{
printf("\nOVERFLOW\n");
}
```

```
else
{
printf("\nEnter Data?");
scanf("%d",&item);
ptr->data = item;
if(head == NULL)
{
head = ptr;
ptr -> next = head;
}
else
{
temp = head;
while(temp -> next != head)
{
temp = temp -> next;
}
temp -> next = ptr;
ptr -> next = head;
}
printf("\nnode inserted\n");
}
}
void begin_delete()
{
struct node *ptr;
if(head == NULL)
{
printf("\nUNDERFLOW");
}
else if(head->next == head)
{
head = NULL;
free(head);
printf("\nnode deleted\n");
}
else
{ ptr = head;
while(ptr -> next != head)
ptr = ptr -> next;
ptr->next = head->next;
free(head);
head = ptr->next;
printf("\nnode deleted\n");
}
}
void last_delete()
{
struct node *ptr, *preptr;
if(head==NULL)
{
```

```
printf("\nUNDERFLOW");
}
else if (head ->next == head)
{
head = NULL;
free(head);
printf("\nnode deleted\n");
}
else
{
ptr = head;
while(ptr ->next != head)
{
preptr=ptr;
ptr = ptr->next;
}
preptr->next = ptr -> next;
free(ptr);
printf("\nnode deleted\n");
}
}
void search()
{
struct node *ptr;
int item,i=0,flag=1;
ptr = head;
if(ptr == NULL)
{
printf("\nEmpty List\n");
}
else
{
printf("\nEnter item which you want to search?\n");
scanf("%d",&item);
if(head ->data == item)
{
printf("item found at location %d",i+1);
flag=0;
}
else
{
while (ptr->next != head)
{
if(ptr->data == item)
{
printf("item found at location %d ",i+1);
flag=0;
break;
}
else
{
```

```
flag=1;
}
i++;
ptr = ptr -> next;
}
}
if(flag != 0)
{
printf("Item not found\n");
}
}
}
void display()
{
struct node *ptr;
ptr=head;
if(head == NULL)
{
printf("\nnothing to print");
}
else
{
printf("\n printing values ... \n");
while(ptr -> next != head)
{
printf("%d\n", ptr -> data);
ptr = ptr -> next;
}
printf("%d\n", ptr -> data);
}
}
```

Output:-

Insertion at beginning:-

```
DEBUG CONSOLE   PROBLEMS 8   OUTPUT   TERMINAL

5.Search for an element
6.Show
7.Exit

Enter your choice?
1

Enter the node data?1

node inserted

*********Main Menu*********

Choose one option from the following list ...

==============================================

1.Insert in begining
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Search for an element
6.Show
7.Exit

Enter your choice?
6

 printing values ...
1
```

Insertion at last:-

```
7.Exit

Enter your choice?
2

Enter Data?2

node inserted

*********Main Menu*********

Choose one option from the following list ...

==============================================

1.Insert in begining
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Search for an element
6.Show
7.Exit

Enter your choice?
6

 printing values ...
1
2
```

Deletion at beginning

```
1.Insert in begining
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Search for an element
6.Show
7.Exit

Enter your choice?
3

node deleted
```

Deletion at last

```
6.Show
7.Exit

Enter your choice?
4

node deleted

*********Main Menu*********

Choose one option from the following list ...

==============================================

1.Insert in begining
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Search for an element
6.Show
7.Exit

Enter your choice?
6

 printing values ...
4
3
2
```

Search

```
1.Insert in begining
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Search for an element
6.Show
7.Exit

Enter your choice?
5

Enter item which you want to search?
3
item found at location 2
*********Main Menu*********
```

7.4.doubly linked circular list

Insertion at beginning:-

```
4.Delete from last
5.Search
6.Show
7.Exit

Enter your choice?
1

Enter Item value1

Node inserted

*********Main Menu*********

Choose one option from the following list ...

==============================================

1.Insert in Beginning
2.Insert at last
3.Delete fromBeginning
4.Delete from last
5.Search
6.Show
7.Exit

Enter your choice?
6

 printing values ...
```

Insertion at last:-

```
Enter your choice?
2

Enter value3

node inserted

*********Main Menu*********

Choose one option from the following list ...

==============================================

1.Insert in Beginning
2.Insert at last
3.Delete fromBeginning
4.Delete from last
5.Search
6.Show
7.Exit

Enter your choice?
6

 printing values ...
1
3
```

Deletion at beginning

```
2.Insert at last
3.Delete fromBeginning
4.Delete from last
5.Search
6.Show
7.Exit

Enter your choice?
3

*********Main Menu*********

Choose one option from the following list ...

==============================================

1.Insert in Beginning
2.Insert at last
3.Delete fromBeginning
4.Delete from last
5.Search
6.Show
7.Exit

Enter your choice?
6

 printing values ...
3
```

Deletion at last

```
OPEN EDITORS
    C infix_podtfix... 1
    C d_link.c c pr... 2
    C list_cir.c c pr... 2
  × C dou_cir_list.... 2
    C simple_list.c... 2
    C infix_postfix... 1
UNTITLED (WORKSPACE)
    d_link.exe
    C dou_cir_list.c 2
    dou_cir_list.exe
    C doubly.c
    doubly.exe
    C linked_list.c
    linked_list.exe
    C list_cir.c 2
    list_cir.exe
    monitor.cpp
OUTLINE
TIMELINE
```

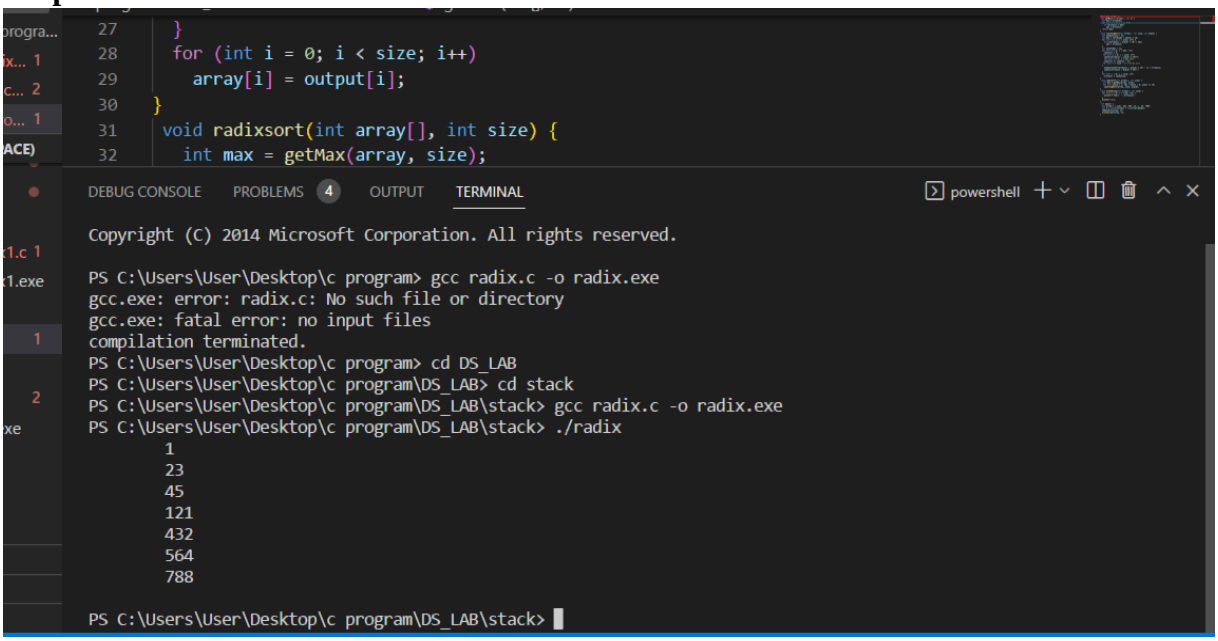
```
6.Show
7.Exit

Enter your choice?
4

node deleted

*********Main Menu*********

Choose one option from the following list ...

==============================================

1.Insert in Beginning
2.Insert at last
3.Delete fromBeginning
4.Delete from last
5.Search
6.Show
7.Exit

Enter your choice?
6

nothing to print
********Main Menu*********

Choose one option from the following list ...
```

8. Write a program in C to perform radix sort
Program:-

```
#include <stdio.h>
int getMax(int array[], int n) {
 int max = array[0];
 for (int i = 1; i < n; i++)
   if (array[i] > max)
     max = array[i];
 return max;
}
void countingSort(int array[], int size, int place) {
 int output[size + 1];
 int max = (array[0] / place) % 10;
 for (int i = 1; i < size; i++) {
   if (((array[i] / place) % 10) > max)
     max = array[i];
 }
 int count[max + 1];
 for (int i = 0; i < max; ++i)
   count[i] = 0;
  for (int i = 0; i < size; i++)
   count[(array[i] / place) % 10]++;
   for (int i = 1; i < 10; i++)
   count[i] += count[i - 1];
  for (int i = size - 1; i >= 0; i--)
  {
   output[count[(array[i] / place) % 10] - 1] = array[i];
   count[(array[i] / place) % 10]--;
 }
 for (int i = 0; i < size; i++)
   array[i] = output[i];
}
void radixsort(int array[], int size) {
  int max = getMax(array, size);
  for (int place = 1; max / place > 0; place *= 10)
   countingSort(array, size, place);
}
void printArray(int array[], int size) {
 for (int i = 0; i < size; ++i) {
   printf("\t%d\n  ", array[i]);
 }
 printf("\n");
}
int main() {
 int array[] = {121, 432, 564, 23, 1, 45, 788};
 int n = sizeof(array) / sizeof(array[0]);
 radixsort(array, n);
 printArray(array, n);
```

```
}
```

Output:-



9. Write a program in C to perform insertion sort

```
#include <stdio.h>
void printArray(int array[], int size) {
 for (int i = 0; i < size; i++) {
   printf("\n\t%d ", array[i]);
 }
 printf("\n");
}
void insertionSort(int array[], int size) {
 for (int step = 1; step < size; step++) {
   int key = array[step];
   int j = step - 1;
while (key < array[j] && j >= 0) {
    array[j + 1] = array[j];
    --j;
   }
   array[j + 1] = key;
 }
}
int main() {
 int data[] = {9, 5, 1, 4, 3,6,45,13,98,8,4,2};
 int size = sizeof(data) / sizeof(data[0]);
 for (int i = 0; i < size; i++) {
   printf("\n\t%d ", data[i]);
 }
 insertionSort(data, size);
 printf("Sorted array in ascending order:\n");
 printArray(data, size);
```

```
}
```

Output:-

```
PS C:\Users\User\Desktop\c program\DS_LAB\stack> gcc insertion.c -o insertion.exe
PS C:\Users\User\Desktop\c program\DS_LAB\stack> ./insertion

        9
        5
        1
        4
        3
        6
        45
        13
        98
        8
        4
        2 Sorted array in ascending order:

        1
        2
        3
        4
        4
        5
        6
        8
        9
        13
        45
        98
PS C:\Users\User\Desktop\c program\DS_LAB\stack>
```

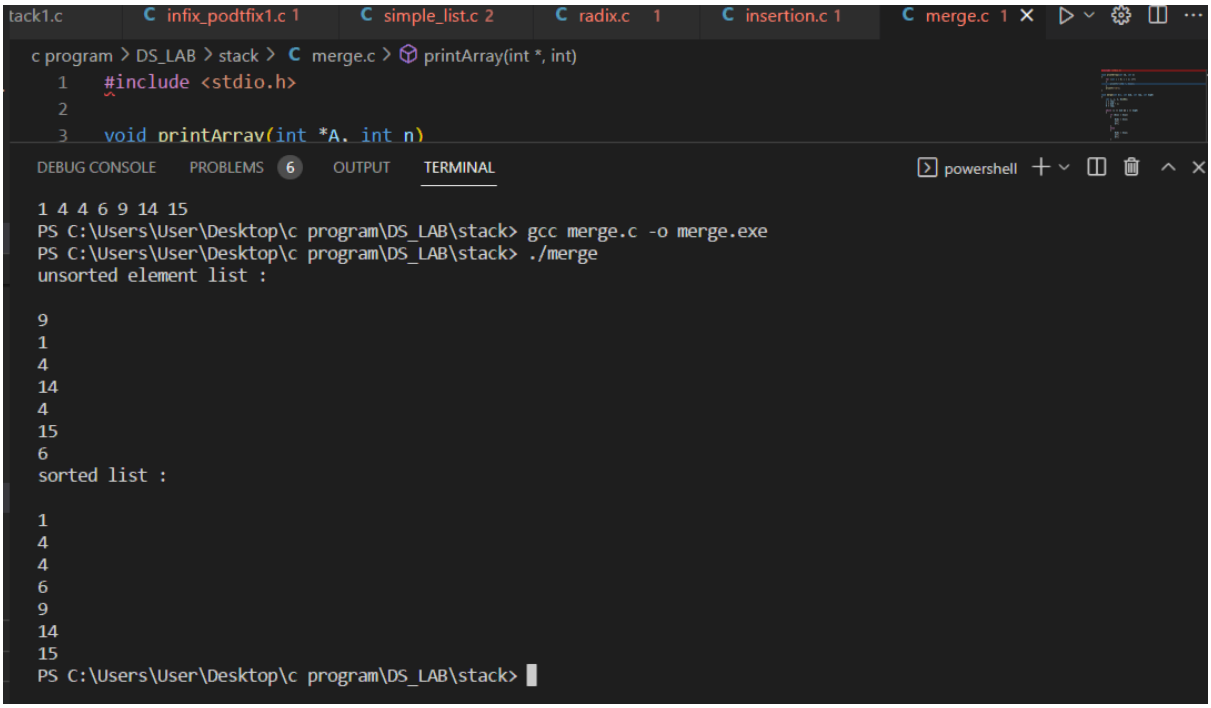
10. Write a program in C to perform merge sort

Program:-

```
#include <stdio.h>
void printArray(int *A, int n)
{
   for (int i = 0; i < n; i++)
   {
      printf("\n%d ", A[i]);
   }
   printf("\n");
}

void merge(int A[], int mid, int low, int high)
{
   int i, j, k, B[100];
   i = low;
   j = mid + 1;
   k = low;

   while (i <= mid && j <= high)
   {
      if (A[i] < A[j])
```

```
        {
            B[k] = A[i];
            i++;
            k++;
        }
        else
        {
            B[k] = A[j];
            j++;
            k++;
        }
    }
    while (i <= mid)
    {
        B[k] = A[i];
        k++;
        i++;
    }
    while (j <= high)
    {
        B[k] = A[j];
        k++;
        j++;
    }
    for (int i = low; i <= high; i++)
    {
        A[i] = B[i];
    }

}

void mergeSort(int A[], int low, int high){
    int mid;
    if(low<high){
        mid = (low + high) /2;
        mergeSort(A, low, mid);
        mergeSort(A, mid+1, high);
        merge(A, mid, low, high);
    }
}

int main()
{
    // int A[] = {9, 14, 4, 8, 7, 5, 6};
    int A[] = {9, 1, 4, 14, 4, 15, 6};
    int n = sizeof(A) / sizeof(A[0]);
   printf("unsorted element list : \n");
    printArray(A, n);
    mergeSort(A, 0, n-1);
    printf("sorted list : \n");
```

```
    printArray(A, n);
    return 0;
}
```



11. Write a program in C to perform selection sort

program:-

```
#include <stdio.h>
void swap(int *xp, int *yp)
{
   int temp = *xp;
   *xp = *yp;
   *yp = temp;
}
void selectionSort(int arr[], int n)
{
   int i, j, min_idx;
 for (i = 0; i < n-1; i++)
   {
      min_idx = i;
      for (j = i+1; j < n; j++)
      if (arr[j] < arr[min_idx])
         min_idx = j;
      if(min_idx != i)
         swap(&arr[min_idx], &arr[i]);
   }
}
 void printArray(int arr[], int size)
```

```
{
   int i;
   for (i=0; i < size; i++)
      printf("%d ", arr[i]);
   printf("\n");
}
 int main()
{
   int arr[] = {64,25,1,4,2,8,2,6,4,12,22,11};
   int n = sizeof(arr)/sizeof(arr[0]);
   selectionSort(arr, n);
   printf("Sorted array: \n");
   printArray(arr, n);
   return 0;
}
```

Output:-