S.No: 13	Exp. Name: ***C program to performs all operations on singly linked list***	Date:2023-06-11

Aim:

Write a program that uses functions to perform the following **operations on singly linked list**

i) Creation

ii) Insertion

iii) Deletion

iv) Traversal

Source Code:

singlelinkedlistalloperations.c

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

}  *head = NULL, *tail = NULL;
void insert();
void Delete();
void display();
void count();
typedef struct node *NODE;
NODE temp, newNode, ptr, ptr2;
int value;
void main() {
   int option = 0;
   printf("Singly Linked List Example - All Operations\n");
   while(1)
{
   printf("Options\n");
   printf("1 : Insert elements into the linked list\n");
   printf("2 : Delete elements from the linked list\n");
   printf("3 : Display the elements in the linked list\n");
   printf("4 : Count the elements in the linked list\n");
   printf("5 : Exit()\n");
   printf("Enter your option : ");
   scanf("%d",&option);
   if (option<=5)  {
      switch(option) {
         case 1:
         insert();
         break;
         case 2:
         Delete();
         break;
         case 3:
         display();
         break;
         case 4:
         count();
         break;
         case 5:
```

```
         exit(0);

      }

      }
      else {
         printf("Enter options from 1 to 5\n");
         break;
      }

}
      }

   void insert() {
         printf("Enter elements for inserting into linked list : ");
         scanf("%d",&value);
         newNode = (NODE) malloc(sizeof(struct node));
         newNode->data = value;
         newNode->next = NULL;
         if(head == NULL) {
            head = newNode;
            tail = newNode;

         }
         else {
            tail->next = newNode;
            tail = newNode;

         }
      }
      void Delete() {
         int i = 1, j = 1, pos, spot, cnt=0;
         temp = head, ptr2 = head;
         while(ptr2!=NULL) {
            cnt++;
            ptr2 = ptr2->next;

         }
         printf("Enter position of the element for deleteing the element : ");
         scanf("%d",&spot);
         while(i<=cnt) {
            if(i == spot) {
               pos = spot;
               break;

            }
            i++;

            }
            if(pos!= spot)
            printf("Invalid position.\n");
            else {
               if(pos==1) {
                  head = head->next;
                  free(temp);
```


ID: 224G1A05C6
Srinivasa Ramanujan Institute of Technology
2022-2026-CSE-B

```
            }

         else {
            while(j<pos) {
               ptr = temp;
               temp = temp->next;
               j++;
            }
            if(temp->next == NULL) {
               ptr->next = NULL;
               free(temp);

            }
            else {
               ptr->next = temp->next;
               free(temp);
            }
         }
         printf("Deleted successfully\n");

      }
}
void display() {
   temp = head;
   printf("The elements in the linked list are : ");
   while(temp!=NULL) {
      printf("%d ",temp->data);
      temp = temp->next;
   }
   printf("\n");

   }
   void count() {
      int count = 0;
      temp = head;
      while(temp!= NULL) {
         count++;
         temp = temp->next;

      }
      printf("No of elements in the linked list are : %d\n",count);

   }
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Singly Linked List Example - All Operations 1
Options 1
1 : Insert elements into the linked list 1
2 : Delete elements from the linked list 1
3 : Display the elements in the linked list 1

ID: 224G1A05C6
2022-2026-CSE-B
Srinivasa Ramanujan Institute of Technology

```
4 : Count the elements in the linked list 1
5 : Exit() 1
Enter your option :  1
Enter elements for inserting into linked list :  111
Options 1
1 : Insert elements into the linked list 1
2 : Delete elements from the linked list 1
3 : Display the elements in the linked list 1
4 : Count the elements in the linked list 1
5 : Exit() 1
Enter your option :  1
Enter elements for inserting into linked list :  222
Options 1
1 : Insert elements into the linked list 1
2 : Delete elements from the linked list 1
3 : Display the elements in the linked list 1
4 : Count the elements in the linked list 1
5 : Exit() 1
Enter your option :  1
Enter elements for inserting into linked list :  333
Options 1
1 : Insert elements into the linked list 1
2 : Delete elements from the linked list 1
3 : Display the elements in the linked list 1
4 : Count the elements in the linked list 1
5 : Exit() 1
Enter your option :  1
Enter elements for inserting into linked list :  444
Options 3
1 : Insert elements into the linked list 3
2 : Delete elements from the linked list 3
3 : Display the elements in the linked list 3
4 : Count the elements in the linked list 3
5 : Exit() 3
Enter your option :  3
The elements in the linked list are : 111 222 333 444  2
Options 2
1 : Insert elements into the linked list 2
2 : Delete elements from the linked list 2
3 : Display the elements in the linked list 2
4 : Count the elements in the linked list 2
5 : Exit() 2
Enter your option :  2
Enter position of the element for deleteing the element :  2
Deleted successfully 3
Options 3
1 : Insert elements into the linked list 3
2 : Delete elements from the linked list 3
3 : Display the elements in the linked list 3
4 : Count the elements in the linked list 3
```

5 : Exit() 3
Enter your option : 3
The elements in the linked list are : 111 333 444 4
Options 4
1 : Insert elements into the linked list 4
2 : Delete elements from the linked list 4
3 : Display the elements in the linked list 4
4 : Count the elements in the linked list 4
5 : Exit() 4
Enter your option : 4
No of elements in the linked list are : 3 5
Options 5
1 : Insert elements into the linked list 5
2 : Delete elements from the linked list 5
3 : Display the elements in the linked list 5
4 : Count the elements in the linked list 5
5 : Exit() 5
Enter your option : 5

Test Case - 2
User Output
Singly Linked List Example - All Operations 1
Options 1
1 : Insert elements into the linked list 1
2 : Delete elements from the linked list 1
3 : Display the elements in the linked list 1
4 : Count the elements in the linked list 1
5 : Exit() 1
Enter your option : 1
Enter elements for inserting into linked list : 001
Options 1
1 : Insert elements into the linked list 1
2 : Delete elements from the linked list 1
3 : Display the elements in the linked list 1
4 : Count the elements in the linked list 1
5 : Exit() 1
Enter your option : 1
Enter elements for inserting into linked list : 010
Options 1
1 : Insert elements into the linked list 1
2 : Delete elements from the linked list 1
3 : Display the elements in the linked list 1
4 : Count the elements in the linked list 1
5 : Exit() 1
Enter your option : 1
Enter elements for inserting into linked list : 100
Options 1
1 : Insert elements into the linked list 1
2 : Delete elements from the linked list 1
3 : Display the elements in the linked list 1

```
4 : Count the elements in the linked list 1
5 : Exit() 1
Enter your option :  1
Enter elements for inserting into linked list :  101
Options 3
1 : Insert elements into the linked list 3
2 : Delete elements from the linked list 3
3 : Display the elements in the linked list 3
4 : Count the elements in the linked list 3
5 : Exit() 3
Enter your option :  3
The elements in the linked list are : 1 10 100 101  2
Options 2
1 : Insert elements into the linked list 2
2 : Delete elements from the linked list 2
3 : Display the elements in the linked list 2
4 : Count the elements in the linked list 2
5 : Exit() 2
Enter your option :  2
Enter position of the element for deleteing the element :  3
Deleted successfully 3
Options 3
1 : Insert elements into the linked list 3
2 : Delete elements from the linked list 3
3 : Display the elements in the linked list 3
4 : Count the elements in the linked list 3
5 : Exit() 3
Enter your option :  3
The elements in the linked list are : 1 10 101  4
Options 4
1 : Insert elements into the linked list 4
2 : Delete elements from the linked list 4
3 : Display the elements in the linked list 4
4 : Count the elements in the linked list 4
5 : Exit() 4
Enter your option :  4
No of elements in the linked list are : 3 5
Options 5
1 : Insert elements into the linked list 5
2 : Delete elements from the linked list 5
3 : Display the elements in the linked list 5
4 : Count the elements in the linked list 5
5 : Exit() 5
Enter your option :  5
```

S.No: 14	Exp. Name: ***C program which performs all operations on double linked list.***	Date:2023-06-15

ID: 224G1A05C6

Aim:

Write a C program that uses functions to perform the following **operations on double linked list**
i) Creation ii) Insertion iii) Deletion iv) Traversal

Source Code:

AllOperationsDLL.c

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>

struct dnode
{
   struct dnode *prev;
   int data;
   struct dnode *next;

};

struct dnode *start = NULL;
void insert(int);
void remov(int);
void display();
int main()
{
   int n,ch;
   do
   {
      printf("Operations on doubly linked list");
      printf("\n1. Insert \n2.Remove\n3. Display\n0. Exit");
      printf("\nEnter Choice 0-4? : ");
      scanf("%d",&ch);
      switch(ch)
      {
         case 1:
         printf("Enter number: ");
         scanf("%d",&n);
         insert(n);
         break;
         case 2:
         printf("Enter number to delete: ");
         scanf("%d",&n);
         remov(n);
         break;
         case 3:
         display();
         break;

      }

   } while (ch != 0);
}
```

```
   void insert(int num)
   {
   struct dnode *nptr,*temp=start;
   nptr = malloc(sizeof(struct dnode));
   nptr->data = num;
   nptr->next = NULL;
   nptr->prev = NULL;

   if(start == NULL)
   {
      start = nptr;
    }
    else
   {
      while(temp->next !=NULL)
       temp = temp->next;
       nptr->prev = temp;
       temp->next = nptr;
      }
    }
    void remov(int num)
    {
      struct dnode *temp = start;
      while(temp != NULL)
    {
     if(temp->data == num)
     {
      if(temp == start)
      {
         start = start->next;
         start->prev->next = NULL;
      }
      else
      {
         if(temp->next == NULL)
         temp->prev->next = NULL;
         else
         {
            temp->prev->next = temp->next;
            temp->next->prev = temp->prev;

         }
         free(temp);
      }
      return ;
   }
   temp = temp->next;
  }
   printf("%d not found.\n",num);
}
void display()
{
   struct dnode *temp = start;
   while(temp != NULL)
   {
   printf("%d\t",temp->data);
```

```
    temp = temp->next;

    }
     printf("\n");
 }
```


Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Operations on doubly linked list 1
1.Insert 1
2.Remove 1
3.Display 1
0.Exit 1
Enter Choice 0-4?: 1
Enter number: 15
Operations on doubly linked list 1
1.Insert 1
2.Remove 1
3.Display 1
0.Exit 1
Enter Choice 0-4?: 1
Enter number: 16
Operations on doubly linked list 1
1.Insert 1
2.Remove 1
3.Display 1
0.Exit 1
Enter Choice 0-4?: 1
Enter number: 17
Operations on doubly linked list 1
1.Insert 1
2.Remove 1
3.Display 1
0.Exit 1
Enter Choice 0-4?: 1
Enter number: 18
Operations on doubly linked list 3
1.Insert 3
2.Remove 3
3.Display 3
0.Exit 3
Enter Choice 0-4?: 3
15 16 17 18 2
Operations on doubly linked list 2
1.Insert 2
2.Remove 2
3.Display 2

```
0.Exit 2
Enter Choice 0-4?:  2
Enter number to delete:  19
19 not found 3
Operations on doubly linked list 3
1.Insert 3
2.Remove 3
3.Display 3
0.Exit 3
Enter Choice 0-4?:  3
15      16      17      18      2
Operations on doubly linked list 2
1.Insert 2
2.Remove 2
3.Display 2
0.Exit 2
Enter Choice 0-4?:  2
Enter number to delete:  16
Operations on doubly linked list 0
1.Insert 0
2.Remove 0
3.Display 0
0.Exit 0
Enter Choice 0-4?:  0
```

S.No: 15	Exp. Name: ***C program to which performs all operations on Circular linked list.***	Date:2023-06-18

Aim:

Write a program that uses functions to perform the following **operations on Circular linked list**
i)Creation ii)insertion iii)deletion iv) Traversal

Source Code:

AlloperationsinCLL.c

```
#include <stdio.h>
#include <stdlib.h>
struct node{
      int data;
         struct node *next;
};
void insert();
void deletion();
void find();
void print();
struct node *head = NULL;
int main()
{
      int choice;
         printf("CIRCULAR LINKED LIST IMPLEMENTATION OF LIST ADT\n");
      while(1)
               {
      printf("1.INSERT ");
      printf("2.DELETE ");
      printf("3.FIND ");
      printf("4.PRINT ");
      printf("5.QUIT\n");
      printf("Enter the choice: ");
      scanf("%d", &choice);

         switch(choice)
      {

      case 1:insert();
      break;
      case 2:deletion();
      break;
      case 3:find();
      break;
      case 4:print();
      break;
      case 5:exit(0);
      break;
   }
}
}
void insert()
{
   int x,n;
   struct node *newnode,*temp = head, *prev;
```


ID: 224G1A05C6
2022-2026-CSE-B
Srinivasa Ramanujan Institute of Technology

```
   newnode = (struct node*)malloc(sizeof(struct node));
   printf("Enter the element to be inserted: ");
      scanf("%d", &x);
      printf("Enter the position of the element: ");
   scanf("%d", &n);
   newnode->data = x;
   newnode->next = NULL;
   if(head == NULL)
   {
    head = newnode;
    newnode->next = newnode;
      }
   else if(n == 1)
   {  temp = head;
   newnode->next = temp;
   while(temp->next != head)
   temp = temp->next;
   temp->next = newnode;
   head = newnode;
   }
else
 {   for(int i=1; i < n-1; i++)
{    temp = temp->next;}
 newnode->next = temp->next;
    temp->next = newnode;
}

}
void deletion()
{
struct node *temp = head, *prev, *temp1 = head;
int key, count = 0;
   printf("Enter the element to be deleted: ");
   scanf("%d", &key);
   if(temp->data == key){
 prev = temp -> next;
 while(temp->next != head)
    {  temp = temp->next;  }
temp->next = prev;
free(head);
 head = prev;
printf("Element deleted\n");
   }
else {
while(temp->next != head)
{     if(temp->data == key){
  count += 1;
     break;

   }
   prev = temp;
   temp = temp->next;
   }
   if(temp->data == key)
   {     prev->next = temp->next;
 free(temp);
```

```
  printf("Element deleted\n");
 }
 else {
    printf("Element does not exist...!\n");
  }
}
}
void find()
{
struct node *temp = head;
int key, count =0;
   printf("Enter the element to be searched: ");
   scanf("%d", &key);
   while(temp->next != head)
{
if(temp->data == key)
   {
   count = 1;
   break;   }
   temp = temp->next;

   }
   if(count == 1)
   printf("Element exist...!\n");
   else
   {   if(temp->data == key)
   printf("Element exist...!\n");
   else
     printf("Element does not exist...!\n");
     }
}
void print() {
   struct node *temp = head;
    printf("The list element are: ");
   while(temp->next != head)
   {
   printf("%d -> ",temp->data);
   temp = temp->next;
   }
   printf("%d -> ", temp->data);
   printf("\n");
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
CIRCULAR LINKED LIST IMPLEMENTATION OF LIST ADT 1
1.INSERT 2.DELETE 3.FIND 4.PRINT 5.QUIT 1
Enter the choice: 1
Enter the element to be inserted: 12

ID: 224G1A05C6

```
Enter the position of the element:  1
1.INSERT 2.DELETE 3.FIND 4.PRINT 5.QUIT 1
Enter the choice:  1
Enter the element to be inserted:  14
Enter the position of the element:  2
1.INSERT 2.DELETE 3.FIND 4.PRINT 5.QUIT 1
Enter the choice:  1
Enter the element to be inserted:  15
Enter the position of the element:  3
1.INSERT 2.DELETE 3.FIND 4.PRINT 5.QUIT 4
Enter the choice:  4
The list element are: 12 -> 14 -> 15 ->  2
1.INSERT 2.DELETE 3.FIND 4.PRINT 5.QUIT 2
Enter the choice:  2
Enter the element to be deleted:  14
Element deleted 4
1.INSERT 2.DELETE 3.FIND 4.PRINT 5.QUIT 4
Enter the choice:  4
The list element are: 12 -> 15 ->  3
1.INSERT 2.DELETE 3.FIND 4.PRINT 5.QUIT 3
Enter the choice:  3
Enter the element to be searched:  12
Element exist...! 5
1.INSERT 2.DELETE 3.FIND 4.PRINT 5.QUIT 5
Enter the choice:  5
```

Test Case - 2

User Output

```
CIRCULAR LINKED LIST IMPLEMENTATION OF LIST ADT 1
1.INSERT 2.DELETE 3.FIND 4.PRINT 5.QUIT 1
Enter the choice:  1
Enter the element to be inserted:  54
Enter the position of the element:  1
1.INSERT 2.DELETE 3.FIND 4.PRINT 5.QUIT 2
Enter the choice:  2
Enter the element to be deleted:  1
Element does not exist...! 4
1.INSERT 2.DELETE 3.FIND 4.PRINT 5.QUIT 4
Enter the choice:  4
The list element are: 54 ->  1
1.INSERT 2.DELETE 3.FIND 4.PRINT 5.QUIT 1
Enter the choice:  1
Enter the element to be inserted:  65
Enter the position of the element:  2
1.INSERT 2.DELETE 3.FIND 4.PRINT 5.QUIT 4
Enter the choice:  4
The list element are: 54 -> 65 ->  5
1.INSERT 2.DELETE 3.FIND 4.PRINT 5.QUIT 5
Enter the choice:  5
```

S.No: 16	Exp. Name: ***Implementation of Circular Queue using Dynamic Array***	Date:2023-06-18

Aim:

Write a program to implement `circular queue` using **dynamic array**.

ID: 224G1A05C6

2022-2026-CSE-B

Srinivasa Ramanujan Institute of Technology

```
Sample Input and Output:
    Enter the maximum size of the circular queue : 3
    1.Enqueue 2.Dequeue 3.Display 4.Exit
    Enter your option : 2
    Circular queue is underflow.
    1.Enqueue 2.Dequeue 3.Display 4.Exit
    Enter your option : 3
    Circular queue is empty.
    1.Enqueue 2.Dequeue 3.Display 4.Exit
    Enter your option : 1
    Enter element : 111
    Successfully inserted.
    1.Enqueue 2.Dequeue 3.Display 4.Exit
    Enter your option : 1
    Enter element : 222
    Successfully inserted.
    1.Enqueue 2.Dequeue 3.Display 4.Exit
    Enter your option : 1
    Enter element : 333
    Successfully inserted.
    1.Enqueue 2.Dequeue 3.Display 4.Exit
    Enter your option : 1
    Enter element : 444
    Circular queue is overflow.
    1.Enqueue 2.Dequeue 3.Display 4.Exit
    Enter your option : 3
    Elements in the circular queue : 111 222 333
    1.Enqueue 2.Dequeue 3.Display 4.Exit
    Enter your option : 2
    Deleted element = 111
    1.Enqueue 2.Dequeue 3.Display 4.Exit
    Enter your option : 1
    Enter element : 444
    Successfully inserted.
    1.Enqueue 2.Dequeue 3.Display 4.Exit
    Enter your option : 3
    Elements in the circular queue : 222 333 444
    1.Enqueue 2.Dequeue 3.Display 4.Exit
    Enter your option : 2
    Deleted element = 222
    1.Enqueue 2.Dequeue 3.Display 4.Exit
    Enter your option : 2
    Deleted element = 333
    1.Enqueue 2.Dequeue 3.Display 4.Exit
    Enter your option : 2
    Deleted element = 444
    1.Enqueue 2.Dequeue 3.Display 4.Exit
    Enter your option : 3
    Circular queue is empty.
    1.Enqueue 2.Dequeue 3.Display 4.Exit
    Enter your option : 4
```

Source Code:

CQueueUsingDynamicArray.c

```
#include <stdio.h>
#include <stdlib.h>
int *cqueue;
int front, rear;
int maxSize;
void initCircularQueue()
{
   cqueue = (int *)malloc(maxSize * sizeof(int));
   front = -1;
   rear = -1;

}
void dequeue()
{
   if (front == -1)
   {
      printf("Circular queue is underflow.\n");
      }
      else
      {
         printf("Deleted element = %d\n", *(cqueue + front));
         if (rear == front) {
            rear = front = -1;
            }
            else if (front == maxSize - 1)
            {
               front = 0;
               } else
               {
                  front++;
                  }  }  }
                  void enqueue(int x)
                  {
               if (((rear == maxSize - 1) && (front == 0)) || (rear + 1 == front))
                  {
            printf("Circular queue is overflow.\n");
            } else
            {
               if (rear == maxSize -1)
               {
                  rear = -1;
                  } else if (front == -1)
                  {
                     front = 0;
                     }
                     rear++;
                     cqueue[rear] = x;
                     printf("Successfully inserted.\n");
               }


                  }
                  void display()
                  {
                     int i;
```

```
   if (front == -1 && rear == -1)
   {
      printf("Circular queue is empty.\n");
      }
      else
      {
         printf("Elements in the circular queue : ");
         if (front <= rear)
         {
            for (i = front; i <= rear; i++)
            {
printf("%d ", *(cqueue + i));
}
} else
{
   for(i = front; i <= maxSize -1; i++)
   {
      printf("%d ", *(cqueue + i));
}
for(i = 0;i <= rear; i++)
{
printf("%d ", *(cqueue + i));
}  }
printf("\n");  }

}
int main()
{
   int op, x;
   printf("Enter the maximum size of the circular queue : ");
   scanf("%d", &maxSize);  initCircularQueue();
   while(1)
   {
printf("1.Enqueue 2.Dequeue 3.Display 4.Exit\n");
printf("Enter your option : ");
scanf("%d",&op);
switch(op)
{
   case 1:
   printf("Enter element : ");
   scanf("%d",&x);
   enqueue(x);
   break;
   case 2:
   dequeue();
   break;
   case 3:
   display();
   break;
   case 4:
   exit(0);
   }  }

}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the maximum size of the circular queue : 3
1.Enqueue 2.Dequeue 3.Display 4.Exit 2
Enter your option : 2
Circular queue is underflow. 3
1.Enqueue 2.Dequeue 3.Display 4.Exit 3
Enter your option : 3
Circular queue is empty. 1
1.Enqueue 2.Dequeue 3.Display 4.Exit 1
Enter your option : 1
Enter element : 111
Successfully inserted. 1
1.Enqueue 2.Dequeue 3.Display 4.Exit 1
Enter your option : 1
Enter element : 222
Successfully inserted. 1
1.Enqueue 2.Dequeue 3.Display 4.Exit 1
Enter your option : 1
Enter element : 333
Successfully inserted. 1
1.Enqueue 2.Dequeue 3.Display 4.Exit 1
Enter your option : 1
Enter element : 444
Circular queue is overflow. 3
1.Enqueue 2.Dequeue 3.Display 4.Exit 3
Enter your option : 3
Elements in the circular queue : 111 222 333 2
1.Enqueue 2.Dequeue 3.Display 4.Exit 2
Enter your option : 2
Deleted element = 111 1
1.Enqueue 2.Dequeue 3.Display 4.Exit 1
Enter your option : 1
Enter element : 444
Successfully inserted. 3
1.Enqueue 2.Dequeue 3.Display 4.Exit 3
Enter your option : 3
Elements in the circular queue : 222 333 444 2
1.Enqueue 2.Dequeue 3.Display 4.Exit 2
Enter your option : 2
Deleted element = 222 2
1.Enqueue 2.Dequeue 3.Display 4.Exit 2
Enter your option : 2
Deleted element = 333 2
1.Enqueue 2.Dequeue 3.Display 4.Exit 2
Enter your option : 2
Deleted element = 444 3
1.Enqueue 2.Dequeue 3.Display 4.Exit 3

```
Enter your option :  3
Circular queue is empty. 4
1.Enqueue 2.Dequeue 3.Display 4.Exit 4
Enter your option :  4
```