Student Name:	Saloni Vishwakarma
Roll No:	13
Practical No:	04
Aim:	Represent a node of a singly linked linear list. Implement the following functions. 1) Create a list 2) Insert an element – at the beginning, at the end and at a specifiedposition in the list 3)Delete an element from the beginning, end or a specified position at thelist 4) Reverse the list 5) Search for an element in the list. Create a menu-driven program to testall the functions

Source Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <malloc.h>
struct node
{
  int data;
  struct node *next;
};
struct node *start = NULL;
struct node *create_ll (struct node *);
struct node *display (struct node *);
struct node *insert_beg (struct node *);
struct node *insert_end (struct node *);
struct node *insert_before (struct node *);
struct node *insert_after (struct node *);
struct node *delete_beg (struct node *);
struct node *delete_end (struct node *);
struct node *delete_bnode (struct node *);
struct node *delete_after (struct node *);
struct node *reverselist (struct node *);
struct node *searchelement (struct node *);

intmain (int argc, char *argv[])
{
  int option;
  do
    {
      printf ("\n\n *****MAIN MENU *****");
      printf ("\n 1: Create a list");
      printf ("\n 2: Display the list");
      printf ("\n 3: Add a node at the beginning");
      printf ("\n 4: Add a node at the end");
      printf ("\n 5: Add a node before a given node");
      printf ("\n 6: Add a node after a given node");
```

```
    printf ("\n 7: Delete a node from the beginning");
    printf ("\n 8: Delete a node from the end");
    printf ("\n 9: Delete a given node");
    printf ("\n 10: Delete a node after a given node");
    printf ("\n 11: Reverse the list");
    printf ("\n 12: Searching an element");
    printf ("\n 13: EXIT");
    printf ("\n Enter your option :");
    scanf ("%d", &option);
    switch (option)
        {
        case 1:
          start = create_ll (start);
          printf ("\n LINKED LIST CREATED");
          break;
        case 2:
          start = display (start);
          break;
        case 3:
          start = insert_beg (start);
          break;
        case 4:
          start = insert_end (start);
          break;
        case 5:
          start = insert_before (start);
          break;
        case 6:
          start = insert_after (start);
          break;
        case 7:
          start = delete_beg (start);
          break;
        case 8:
          start = delete_end (start);
          break;
        case 9:
          start = delete_bnode (start);
          break;
        case 10:
          start = delete_after (start);
          break;
        case 11:
          start = reverselist (start);
          break;
        case 12:
          start = searchelement (start);
          break;
        }
  }
while (option != 13);
getch ();
```

```
 return 0;
}

struct node *create_ll (struct node *start)
{
 struct node *new_node, *ptr;
 int num;
 printf ("Enter -1 to end\n");
 printf ("Enter the data :");
 scanf ("%d", &num);
 while (num != -1)
  {
   new_node = (struct node *) malloc (sizeof (struct node));
   new_node->data = num;
   if (start == NULL)
      {
       new_node->next = NULL;
       start = new_node;
      }
   else
      {
       ptr = start;
       while (ptr->next != NULL)
         ptr = ptr->next;
       ptr->next = new_node;
       new_node->next = NULL;
      }
   printf ("\n Enter the data : ");
   scanf ("%d", &num);
  }
 return start;
}

struct node *display (struct node *start)
{
 struct node *ptr;
 ptr = start;
 while (ptr != NULL)
  {
   printf ("\t %d", ptr->data);
   ptr = ptr->next;
  }
 return start;
}

struct node *insert_beg (struct node *start)
{
 struct node *new_node;
 int num;
 printf ("\n Enter the data : ");
 scanf ("%d", &num);
 new_node = (struct node *) malloc (sizeof (struct node));
```

```
 new_node->data = num;
 new_node->next = start;
 start = new_node;
 return start;
}

struct node *insert_end (struct node *start)
{
 struct node *ptr, *new_node;
 int num;
 printf ("\n Enter the data : ");
 scanf ("%d", &num);
 new_node = (struct node *) malloc (sizeof (struct node));
 new_node->data = num;
 new_node->next = NULL;
 ptr = start;
 while (ptr->next != NULL)
   ptr = ptr->next;
 ptr->next = new_node;
 return start;
}

struct node *insert_before (struct node *start)
{
 struct node *new_node, *ptr, *preptr;
 int num, val;
 printf ("\n Enter the data : ");
 scanf ("%d", &num);
 printf ("\n Enter the value before which the data has to be inserted : ");
 scanf ("%d", &val);
 new_node = (struct node *) malloc (sizeof (struct node));
 new_node->data = num;
 ptr = start;
 while (ptr->data != val)
   {
    preptr = ptr;
    ptr = ptr->next;
   }
 preptr->next = new_node;
 new_node->next = ptr;
 return start;
}

struct node *insert_after (struct node *start)
{
 struct node *new_node, *ptr, *preptr;
 int num, val;
 printf ("\n Enter the data : ");
 scanf ("%d", &num);
 printf ("\n Enter the value after which the data has to be inserted : ");
 scanf ("%d", &val);
 new_node = (struct node *) malloc (sizeof (struct node));
```

```
  new_node->data = num;
  ptr = start;
  preptr = ptr;
  while (preptr->data != val)
    {
      preptr = ptr;
      ptr = ptr->next;
    }
  preptr->next = new_node;
  new_node->next = ptr;
  return start;
}

struct node *delete_beg (struct node *start)
{
  struct node *ptr;
  ptr = start;
  start = start->next;
  free (ptr);
  return start;
}

struct node *delete_end (struct node *start)
{
  struct node *ptr, *preptr;
  ptr = start;
  while (ptr->next != NULL)
    {
      preptr = ptr;
      ptr = ptr->next;
    }
  preptr->next = NULL;
  free (ptr);
  return start;
}

struct node *delete_bnode (struct node *start)
{
  struct node *ptr, *preptr;
  int val;
  printf ("\n Enter the value of the node which has to be deleted : ");
  scanf ("%d", &val);
  ptr = start;
  preptr = ptr;
  if (preptr->data == val)
    {
      start = delete_beg (start);
      return start;
    }
  else
    {
      while (ptr->data != val)
```

```
        {
          preptr = ptr;
          ptr = ptr->next;
        }
     preptr->next = ptr->next;
     free (ptr);
     return start;
   }
}

struct node *delete_after (struct node *start)
{
  struct node *ptr, *preptr;
  int val;
  printf ("\n Enter the value of the node which has to be deleted : ");
  scanf ("%d", &val);
  ptr = start;
  if (ptr->data == val)
    {
     start = delete_beg (start);
     return start;
    }
  else
    {
     while (preptr->data != val)
         {
          preptr = ptr;
          ptr = ptr->next;
         }
     preptr->next = ptr->next;
     free (ptr);
     return start;
    }
}

struct node *reverselist (struct node *start)
{
  struct node *prevNode, *curNode;
  if (start != NULL)
    {
     prevNode = start;
     curNode = start->next;
     start = start->next;
     prevNode->next = NULL;
     while (start != NULL)
         {
          start = start->next;
          curNode->next = prevNode;
          prevNode = curNode;
          curNode = start;
         }
     start = prevNode;
```

```
    printf ("SUCCESSFULLY REVERSED LIST\n");
    struct node *temp;
    if (start == NULL)
        {
         printf ("List is empty.");
        }
    else
        {
         temp = start;
         printf ("List is: \n");
         while (temp != NULL)
          {
           printf ("%d\n", temp->data);
           temp = temp->next;
          }
        }
   }
  return start;
 }

 struct node *searchelement (struct node *start)
 {
  int searchval;
  printf ("Enter the element to be searched: \n");
  scanf ("%d", &searchval);
  struct node *ptr = start;
  int f=0;
  while (ptr != NULL)
   {
    if (ptr->data = searchval)
        {
         f=1;
         break;
        }
    else
        {
          ptr = ptr->next;
        }
   }
   if(f==1)
   printf ("Value found\n");
   else
   printf("Value not found\n");
  return start;
}
```

Output:

1. Create a list:

```
*****MAIN MENU *****
1: Create a list
2: Display the list
3: Add a node at the beginning
4: Add a node at the end
5: Add a node before a given node
6: Add a node after a given node
7: Delete a node from the beginning
8: Delete a node from the end
9: Delete a given node
10: Delete a node after a given node
11: Reverse the list
12: Searching an element
13: EXIT
Enter your option :1
Enter -1 to end
Enter the data :7

Enter the data : 8

Enter the data : 9

Enter the data : -1

LINKED LIST CREATED
```

```
*****MAIN MENU *****
1: Create a list
2: Display the list
3: Add a node at the beginning
4: Add a node at the end
5: Add a node before a given node
6: Add a node after a given node
7: Delete a node from the beginning
8: Delete a node from the end
9: Delete a given node
10: Delete a node after a given node
11: Reverse the list
12: Searching an element
13: EXIT
Enter your option :2
        7       8       9
```

2. Insert a node at the beginning:

```
*****MAIN MENU *****
1: Create a list
2: Display the list
3: Add a node at the beginning
4: Add a node at the end
5: Add a node before a given node
6: Add a node after a given node
7: Delete a node from the beginning
8: Delete a node from the end
9: Delete a given node
10: Delete a node after a given node
11: Reverse the list
12: Searching an element
13: EXIT
Enter your option :3

Enter the data : 6


*****MAIN MENU *****
1: Create a list
2: Display the list
3: Add a node at the beginning
4: Add a node at the end
5: Add a node before a given node
6: Add a node after a given node
7: Delete a node from the beginning
8: Delete a node from the end
9: Delete a given node
10: Delete a node after a given node
11: Reverse the list
12: Searching an element
13: EXIT
Enter your option :2
        6       7       8       9
```

3. Insert a node at the end:

```
*****MAIN MENU *****
1: Create a list
2: Display the list
3: Add a node at the beginning
4: Add a node at the end
5: Add a node before a given node
6: Add a node after a given node
7: Delete a node from the beginning
8: Delete a node from the end
9: Delete a given node
10: Delete a node after a given node
11: Reverse the list
12: Searching an element
13: EXIT
Enter your option :4

Enter the data : 10


*****MAIN MENU *****
1: Create a list
2: Display the list
3: Add a node at the beginning
4: Add a node at the end
5: Add a node before a given node
6: Add a node after a given node
7: Delete a node from the beginning
8: Delete a node from the end
9: Delete a given node
10: Delete a node after a given node
11: Reverse the list
12: Searching an element
13: EXIT
Enter your option :2
        6       7       8       9       10
```

4. Insert a node before a given node:

```
*****MAIN MENU *****
1: Create a list
2: Display the list
3: Add a node at the beginning
4: Add a node at the end
5: Add a node before a given node
6: Add a node after a given node
7: Delete a node from the beginning
8: Delete a node from the end
9: Delete a given node
10: Delete a node after a given node
11: Reverse the list
12: Searching an element
13: EXIT
Enter your option :5

Enter the data : 78

Enter the value before which the data has to be inserted : 8


*****MAIN MENU *****
1: Create a list
2: Display the list
3: Add a node at the beginning
4: Add a node at the end
5: Add a node before a given node
6: Add a node after a given node
7: Delete a node from the beginning
8: Delete a node from the end
9: Delete a given node
10: Delete a node after a given node
11: Reverse the list
12: Searching an element
13: EXIT
Enter your option :2
        6       7       78      8       9       10
```

5. Insert a node after a given node:

```
*****MAIN MENU *****
1: Create a list
2: Display the list
3: Add a node at the beginning
4: Add a node at the end
5: Add a node before a given node
6: Add a node after a given node
7: Delete a node from the beginning
8: Delete a node from the end
9: Delete a given node
10: Delete a node after a given node
11: Reverse the list
12: Searching an element
13: EXIT
Enter your option :6

Enter the data : 89

Enter the value after which the data has to be inserted : 8
```

```
*****MAIN MENU *****
1: Create a list
2: Display the list
3: Add a node at the beginning
4: Add a node at the end
5: Add a node before a given node
6: Add a node after a given node
7: Delete a node from the beginning
8: Delete a node from the end
9: Delete a given node
10: Delete a node after a given node
11: Reverse the list
12: Searching an element
13: EXIT
Enter your option :2
        6       7       78      8       89      9       10
```

6. Delete a node from the beginning:

```
*****MAIN MENU *****
1: Create a list
2: Display the list
3: Add a node at the beginning
4: Add a node at the end
5: Add a node before a given node
6: Add a node after a given node
7: Delete a node from the beginning
8: Delete a node from the end
9: Delete a given node
10: Delete a node after a given node
11: Reverse the list
12: Searching an element
13: EXIT
Enter your option :7


*****MAIN MENU *****
1: Create a list
2: Display the list
3: Add a node at the beginning
4: Add a node at the end
5: Add a node before a given node
6: Add a node after a given node
7: Delete a node from the beginning
8: Delete a node from the end
9: Delete a given node
10: Delete a node after a given node
11: Reverse the list
12: Searching an element
13: EXIT
Enter your option :2
        7       78      8       89      9       10
```

7. Delete a node at the end:

```
*****MAIN MENU *****
1: Create a list
2: Display the list
3: Add a node at the beginning
4: Add a node at the end
5: Add a node before a given node
6: Add a node after a given node
7: Delete a node from the beginning
8: Delete a node from the end
9: Delete a given node
10: Delete a node after a given node
11: Reverse the list
12: Searching an element
13: EXIT
Enter your option :8


*****MAIN MENU *****
1: Create a list
2: Display the list
3: Add a node at the beginning
4: Add a node at the end
5: Add a node before a given node
6: Add a node after a given node
7: Delete a node from the beginning
8: Delete a node from the end
9: Delete a given node
10: Delete a node after a given node
11: Reverse the list
12: Searching an element
13: EXIT
Enter your option :2
        7       78      8       89      9
```

8. Delete a given node:

```
*****MAIN MENU *****
1: Create a list
2: Display the list
3: Add a node at the beginning
4: Add a node at the end
5: Add a node before a given node
6: Add a node after a given node
7: Delete a node from the beginning
8: Delete a node from the end
9: Delete a given node
10: Delete a node after a given node
11: Reverse the list
12: Searching an element
13: EXIT
Enter your option :9

Enter the value of the node which has to be deleted : 78


*****MAIN MENU *****
1: Create a list
2: Display the list
3: Add a node at the beginning
4: Add a node at the end
5: Add a node before a given node
6: Add a node after a given node
7: Delete a node from the beginning
8: Delete a node from the end
9: Delete a given node
10: Delete a node after a given node
11: Reverse the list
12: Searching an element
13: EXIT
Enter your option :2
        7       8       89      9
```

9. Delete a node after a given node:

```
*****MAIN MENU *****
1: Create a list
2: Display the list
3: Add a node at the beginning
4: Add a node at the end
5: Add a node before a given node
6: Add a node after a given node
7: Delete a node from the beginning
8: Delete a node from the end
9: Delete a given node
10: Delete a node after a given node
11: Reverse the list
12: Searching an element
13: EXIT
Enter your option :10

Enter the value of the node which has to be deleted : 8


*****MAIN MENU *****
1: Create a list
2: Display the list
3: Add a node at the beginning
4: Add a node at the end
5: Add a node before a given node
6: Add a node after a given node
7: Delete a node from the beginning
8: Delete a node from the end
9: Delete a given node
10: Delete a node after a given node
11: Reverse the list
12: Searching an element
13: EXIT
Enter your option :2
        7       8       9
```

10. Reverse the list:

```
*****MAIN MENU *****
1: Create a list
2: Display the list
3: Add a node at the beginning
4: Add a node at the end
5: Add a node before a given node
6: Add a node after a given node
7: Delete a node from the beginning
8: Delete a node from the end
9: Delete a given node
10: Delete a node after a given node
11: Reverse the list
12: Searching an element
13: EXIT
Enter your option :11
SUCCESSFULLY REVERSED LIST
List is:
9
8
7
```

11. Searching an element:

```
*****MAIN MENU *****
1: Create a list
2: Display the list
3: Add a node at the beginning
4: Add a node at the end
5: Add a node before a given node
6: Add a node after a given node
7: Delete a node from the beginning
8: Delete a node from the end
9: Delete a given node
10: Delete a node after a given node
11: Reverse the list
12: Searching an element
13: EXIT
Enter your option :12
Enter the element to be searched:
5
Value found
```

```
*****MAIN MENU *****
1: Create a list
2: Display the list
3: Add a node at the beginning
4: Add a node at the end
5: Add a node before a given node
6: Add a node after a given node
7: Delete a node from the beginning
8: Delete a node from the end
9: Delete a given node
10: Delete a node after a given node
11: Reverse the list
12: Searching an element
13: EXIT
Enter your option :12
Enter the element to be searched:
8
Value found
```

Result: The concept of Singly Linked List has been studied and various allowable operations of singly linked list have been implemented in C.