ACADEMY OF TECHNOLOGY

NAME : PRIYADARSHAN GHOSH

COLLEGE ROLL NO: 72

UNIVERSITY ROLL NO: 16900319072

DEPARTMENT: ECE-1(Y)

SEMESTER: 3rd

PAPER CODE : ES-CS391

➢ Laboratory *Assignment #10*

WRITE MENU DRIVEN PROGRAM TO PERFORM FOLLOWING OPERATIONS USING FUNCTIONS:

A. CREATION OF SINGLY LINKED LIST

B. DISPLAY OF SINGLY LINKED LIST

C. INSERT A NODE IN DIFFERENT POSITIONS OF SINGLY LINKED LIST

D. DELETE A NODE FROM DIFFERENT POSITIONS OF SINGLY LINKED LIST

Ans:

```
#include <stdio.h>
#include <stdlib.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_node(struct node *);
```

```
        struct node *delete_after(struct node *);
        struct node *delete_list(struct node *);

        int main(int argc, char *argv[])
{
                int option;
                do
{
        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: Delete the entire list");
        printf("\n 13: EXIT");
        printf("\n\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_node(start);
break;

case 10: start = delete_after(start);
```

```
        break;

        case 11: start = delete_list(start);
        printf("\n LINKED LIST DELETED");
        break;

}
}
while(option !=13);

return 0;
}
struct node *create_ll(struct node *start)
{
        struct node *new_node, *ptr;
        int num;
                printf("\n Enter -1 to end");
                printf("\n 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_node(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(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 after which the node has to
deleted : ");
        scanf("%d", &val);
            ptr = start;
        preptr = ptr;
while(preptr -> data != val)
{
        preptr = ptr;
        ptr = ptr -> next;
}
        preptr -> next=ptr -> next;
        free(ptr);
return start;
}
        struct node *delete_list(struct node *start)
{
        struct node *ptr;
        if(start!=NULL)
```

```
{
        ptr=start;
        while(ptr != NULL)
{
        printf("\n %d is to be deleted next”, ptr -> data");
        start = delete_beg(ptr);
        ptr = start;
}

}
        return start;
}
```

OUTPUT =>

```
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: Delete the entire list
13: EXIT
```

Enter your option : 1

Enter -1 to end

Enter the data : 2

Enter the data : 4

Enter the data : 5

Enter the data : 6

Enter the data : 9

Enter the data : -1

LINKED LIST CREATED

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: Delete the entire list

13: EXIT

Enter your option : 2

2 4 5 6 9

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: Delete the entire list

13: EXIT

Enter your option : 3

Enter the data : 12

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: Delete the entire list

13: EXIT

Enter your option : 4

Enter the data : 22

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: Delete the entire list

13: EXIT

Enter your option : 2

12 2 4 5 6 9 22

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: Delete the entire list

13: EXIT

Enter your option : 7

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: Delete the entire list

13: EXIT

Enter your option : 2

2 4 5 6 9 22

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: Delete the entire list

13: EXIT

Enter your option : 8

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: Delete the entire list

13: EXIT

Enter your option : 2

2 4 5 6 9

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: Delete the entire list

13: EXIT

Enter your option : 11

-14992 is to be deleted nextö, ptr -> data

10813440 is to be deleted nextö, ptr -> data

10813440 is to be deleted nextö, ptr -> data

10813440 is to be deleted nextö, ptr -> data

10813440 is to be deleted nextö, ptr -> data

LINKED LIST DELETED

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: Delete the entire list

13: EXIT

Enter your option : 2

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: Delete the entire list

13: EXIT

```
 Enter your option : 13


--------------------------------

Process exited after 56.05 seconds with return value 0

Press any key to continue . . .
```