# **Name: Abdurrahman Qureshi**

# **Roll No: 242466**

Practical No: 3

1) Insertion, deletion and reversing operations with singly linked

CODE

#include <stdio.h>

#include <malloc.h>

#include <stdlib.h>

struct node

{

    int data;

    struct node \*next;

};

struct node \*start = NULL;

void createll();

void Insertbegin();

void Insertend();

void Insertbefore();

void Insertafter();

void DeleteAfter();

void DeleteBegin();

void DeleteEnd();

void display();

int main()

{

    int n;

    while (1)

    {

        printf("\n\nEnter 1 for create linked list\n");

        printf("Enter 2 for insert in the beginning\n ");

        printf("Enter 3 for insert in the end\n ");

        printf("Enter 4 for insert before a given node \n");

        printf("Enter 5 for insert after a given node\n ");

        printf("Enter 6 for deleting at the beginning\n ");

        printf("Enter 7 for delete at the end\n");

        printf("Enter 8 for delete after a given node\n ");

        printf("Enter 9 to display \n");

        printf("ENTER YOUR CHOICE");

        scanf("%d", &n);

        switch (n)

        {

        case 1:

            createll();

            break;

        case 2:

            Insertbegin();

            break;

        case 3:

            Insertend();

            break;

        case 4:

            Insertbefore();

            break;

        case 5:

            Insertafter();

            break;

        case 6:

            DeleteBegin();

            break;

        case 7:

            DeleteEnd();

            break;

        case 8:

            DeleteAfter();

            break;

        case 9:

            display();

            break;

        default:

        {

            printf("Error");

        }

        }

    }

}

void createll()

{

    struct node \*avail, \*ptr;

    avail = (struct node \*)malloc(sizeof(struct node));

    if (avail == NULL)

    {

        printf("\n out of space");

    }

    printf("Enter the data:-\n");

    scanf("%d", &avail->data);

    avail->next = NULL;

    if (start == NULL)

    {

        start = avail;

    }

    else

    {

        while (ptr->next != NULL)

        {

            ptr = ptr->next;

        }

        ptr->next = avail;

    }

}

void display()

{

    struct node \*ptr;

    if (start == NULL)

    {

        printf("\n Linked list is empty");

    }

    else

    {

        ptr = start;

        printf("Linked list elemens are:-");

        while (ptr != NULL)

        {

            printf("%d\t", ptr->data);

            printf("%d\t", ptr->next);

            ptr = ptr->next;

        }

    }

}

void Insertbegin()

{

    struct node \*avail;

    avail = (struct node \*)malloc(sizeof(struct node));

    if (avail == NULL)

    {

        printf("\n out of space");

    }

    printf("Enter the data:-\n");

    scanf("%d", &avail->data);

    avail->next = NULL;

    if (start == NULL)

    {

        start = avail;

    }

    else

    {

        avail->next = start;

        start = avail;

    }

}

void Insertend()

{

    struct node \*avail, \*ptr;

    avail = (struct node \*)malloc(sizeof(struct node));

    if (avail == NULL)

    {

        printf("\n out of space");

    }

    printf("Enter the data:-\n");

    scanf("%d", &avail->data);

    avail->next = NULL;

    ptr = start;

    while (ptr->next != NULL)

    {

        ptr = ptr->next;

    }

    ptr->next = avail;

}

void Insertbefore()

{

    struct node \*avail, \*ptr, \*preptr;

    int val;

    avail = (struct node \*)malloc(sizeof(struct node));

    if (avail == NULL)

    {

        printf("\n out of space");

    }

    printf("Enter the data values for the node:-\n");

    scanf("%d", &avail->data);

    printf("Enter the node value to add before it:-\n");

    scanf("%d", &val);

    avail->next = NULL;

    ptr = start;

    while (ptr->data != val)

    {

        preptr = ptr;

        ptr = ptr->next;

    }

    preptr->next = avail;

    avail->next = ptr;

}

void Insertafter()

{

    struct node \*avail, \*ptr, \*preptr;

    int val;

    avail = (struct node \*)malloc(sizeof(struct node));

    if (avail == NULL)

    {

        printf("\n out of space");

    }

    printf("Enter the data values for the node:-\n");

    scanf("%d", &avail->data);

    printf("Enter the node value to add after it:-\n");

    scanf("%d", &val);

    avail->next = NULL;

    ptr = start;

    while (preptr->data != val)

    {

        preptr = ptr;

        ptr = ptr->next;

    }

    preptr->next = avail;

    avail->next = ptr;

}

void DeleteBegin()

{

    struct node \*ptr;

    if (start == NULL)

    {

        printf("\n Linked list is empty");

    }

    ptr = start;

    start = start->next;

    free(ptr);

}

void DeleteEnd()

{

    struct node \*ptr, \*preptr;

    if (start == NULL)

    {

        printf("Linked list is empty");

    }

    ptr = start;

    while (ptr->next != NULL)

    {

        preptr = ptr;

        ptr = ptr->next;

    }

    preptr->next = NULL;

    free(ptr);

}

void DeleteAfter()

{

    struct node \*ptr, \*preptr;

    int val;

    if (start == NULL)

    {

        printf("Linked list is empty");

    }

    printf("Enter the value to delete node:-\n");

    scanf("%d", &val);

    ptr = start;

    while (ptr->data != val)

    {

        preptr = ptr;

        ptr = ptr->next;

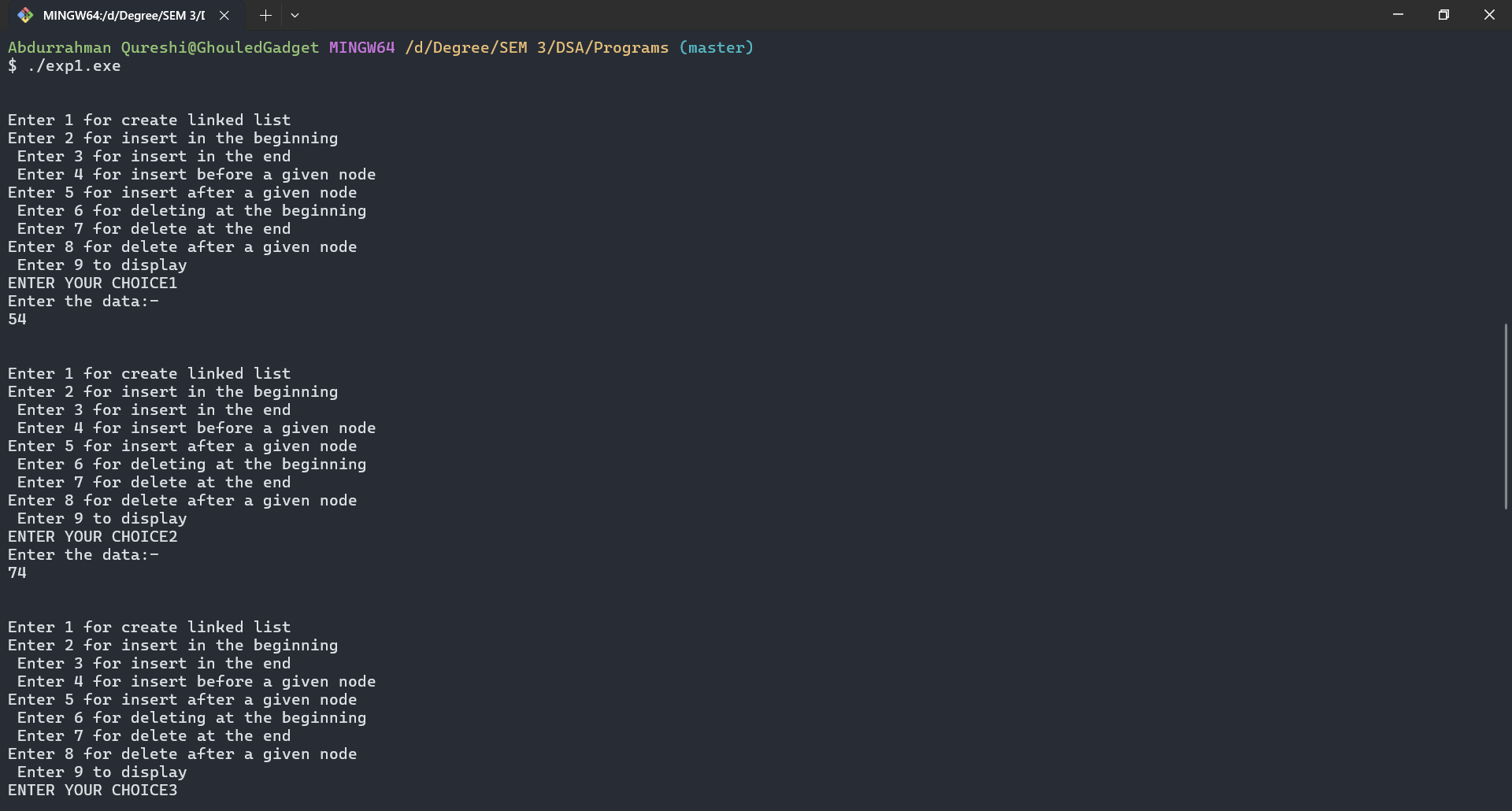
    }

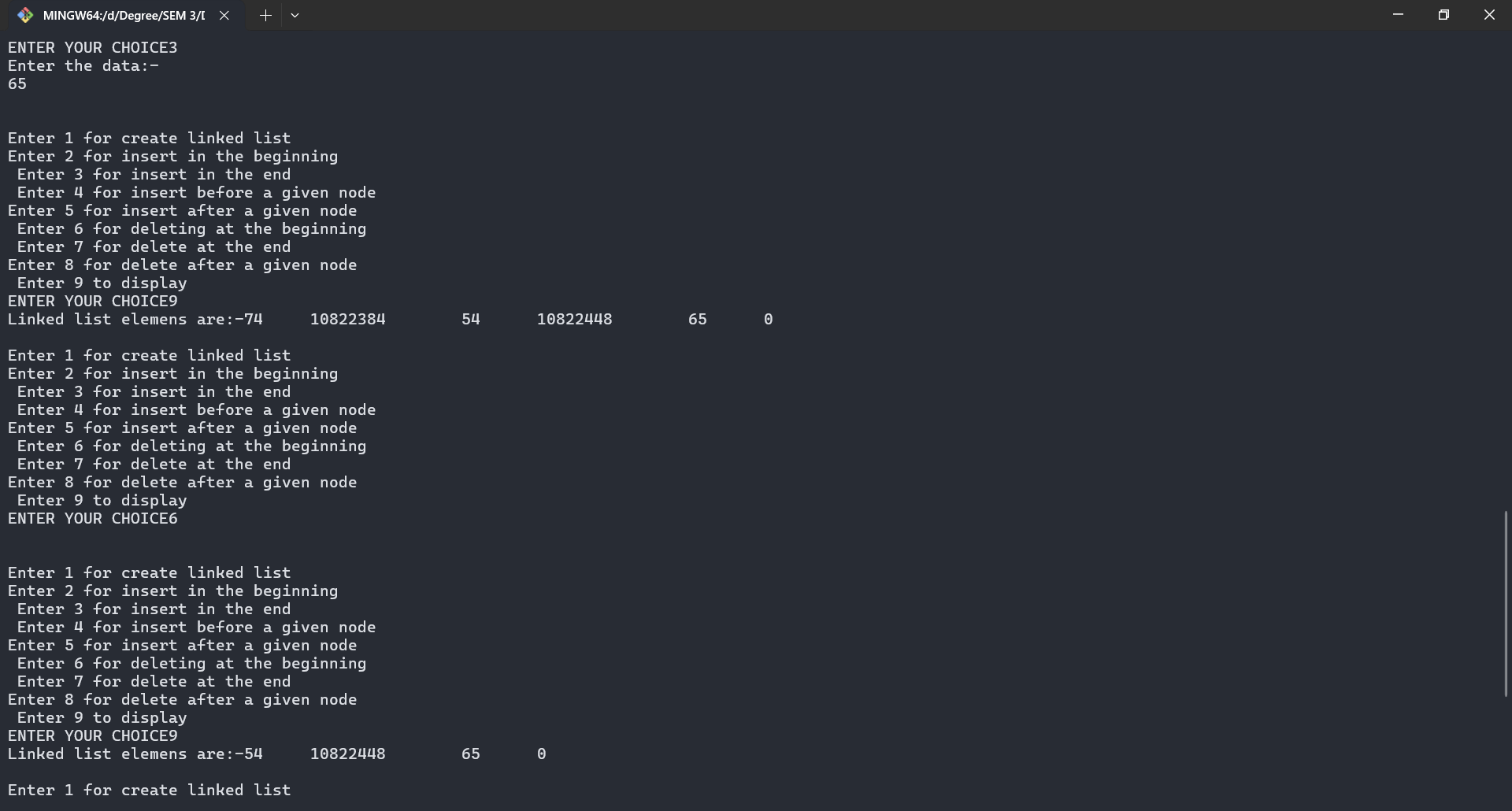
    preptr->next = ptr->next;

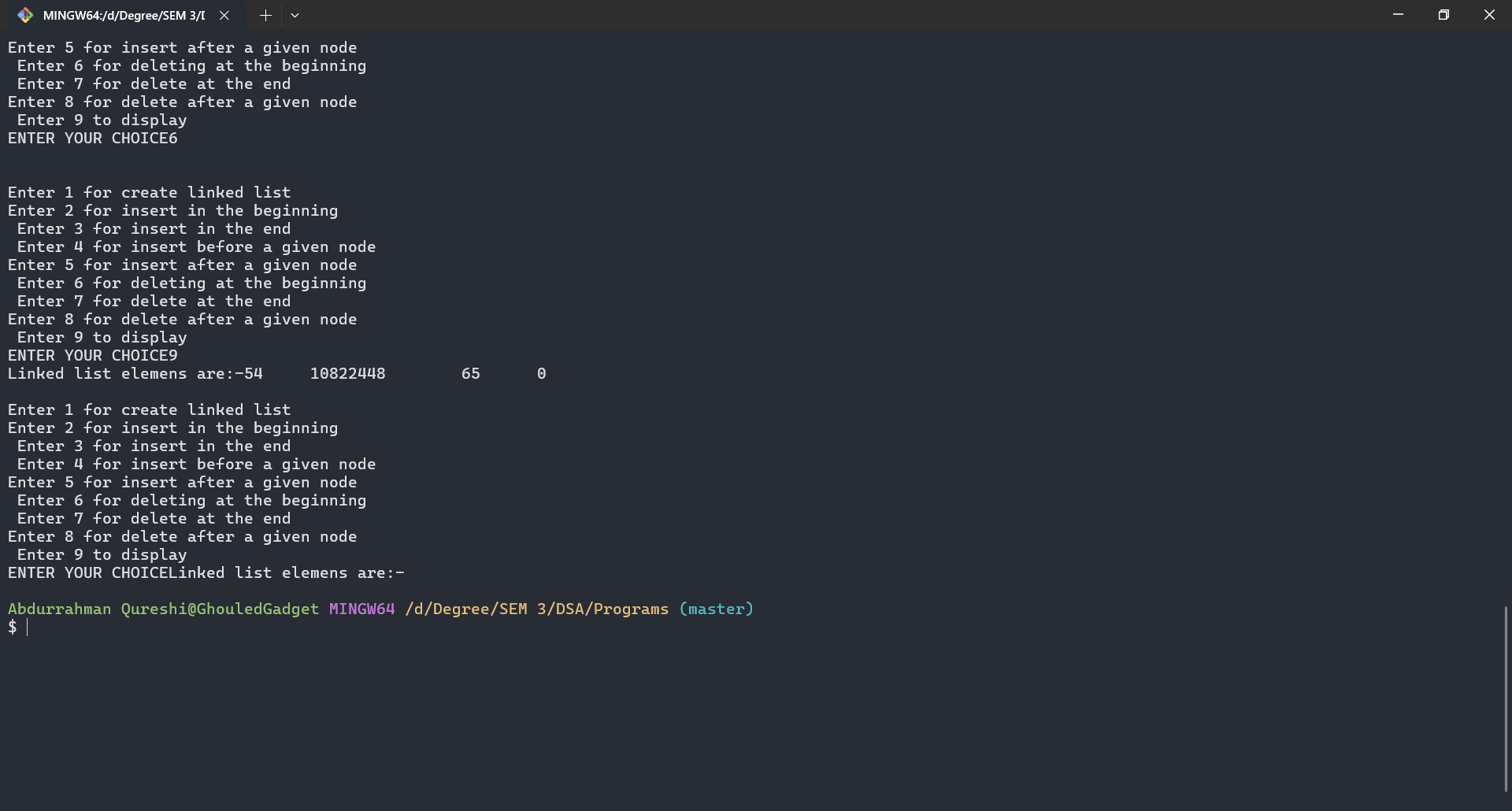
    free(ptr);

}

OUTPUT







Tools used :

Software: Dev c++

Hardware: Lab Computers

References: Mam notes.

Conclusion

* insertBeg(): This function simply inserts an element at the front/beginning of the linked list.
* insertEnd(): This function inserts an element at the end of the linked list.
* deleteBeg(): This function simply deletes an element from the front/beginning of the linked list.
* deleteEnd(): This function simply deletes an element from the end of the linked list.