Data Structures and Algorithms Lab – 3 – L55+L56

Faculty: Prof. Jayasudha M **Work done by:** Arvind CB [19BCE1221]

Write a program to create a singly linked list and implement functions to perform following operations.

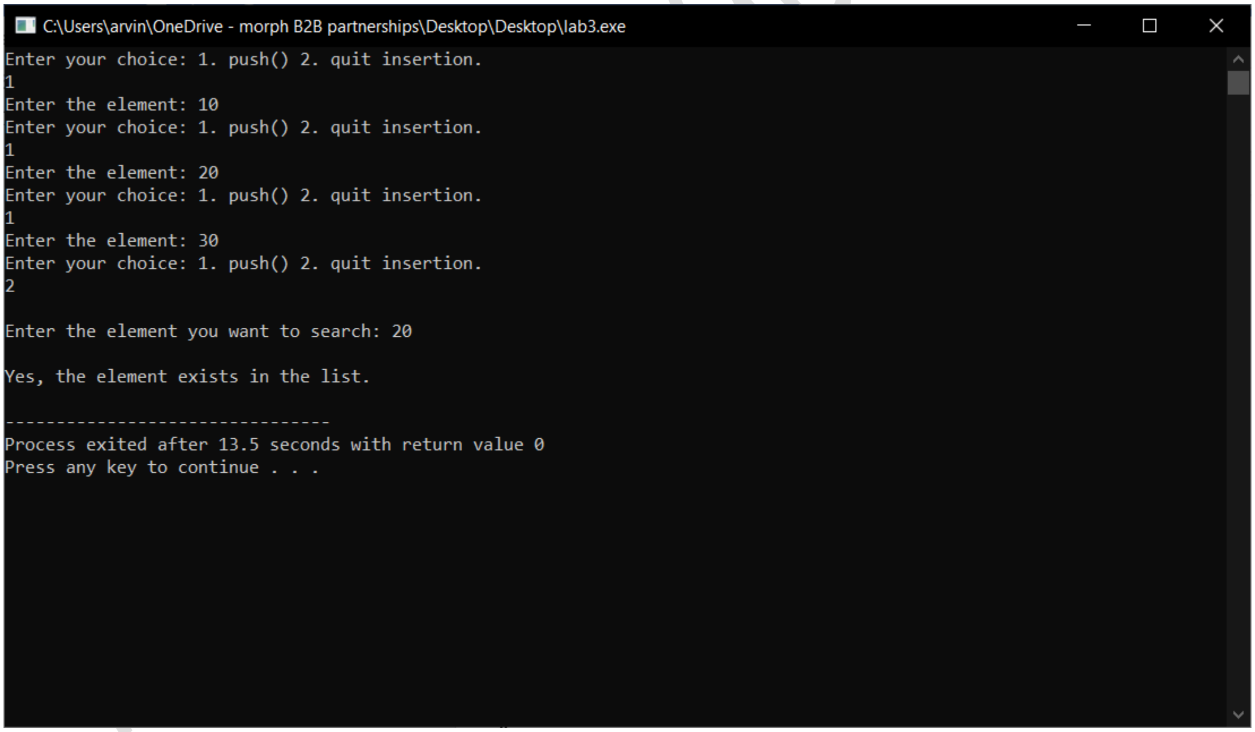
a. Search for a specified value

Code:

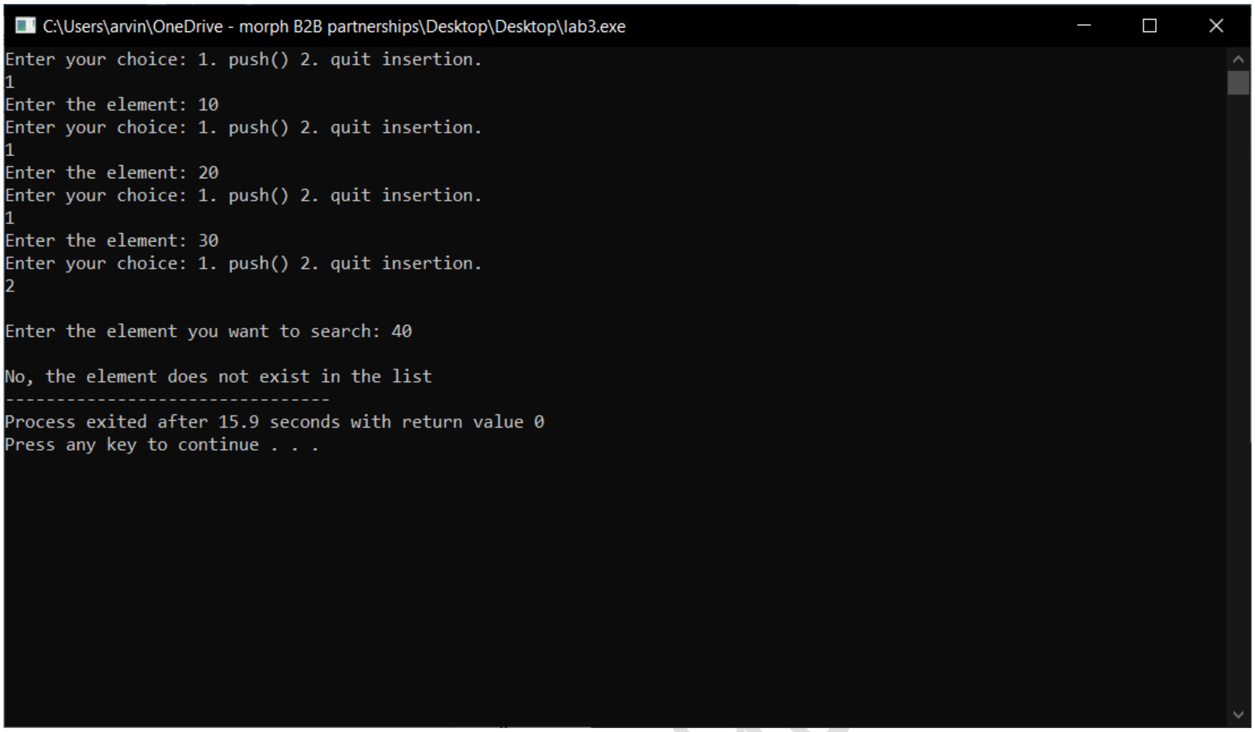
```
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>
struct listNode
{
   int key;
   struct listNode* next;
};
void push(struct listNode** head_ref, int new_key)
{
   struct listNode* new_node = (struct listNode*) malloc(sizeof(struct listNode));
   new_node->key  = new_key;
   new_node->next = (*head_ref);
   (*head_ref)    = new_node;
}
bool search(struct listNode* head, int x)
{
   struct listNode* current = head;
   while (current != NULL)
```

```
    {
        if (current->key == x)
            return true;
        current = current->next;
    }
    return false;
}
int main()
{
    //Option 1
     {
                struct listNode* head = NULL;
         int tosearch, choice;
         do
         {
                printf("Enter your choice: 1. push() 2. quit insertion.\n");
                scanf("%d", &choice);
                switch(choice)
                {
                            case 1:
                                      printf("Enter the element: ");
                                      int element;
                                      scanf("%d", &element);
                                      push(&head, element);
                                      break;
                            case 2:
                                break;

                        }

                }while(choice!=2);

                printf("\nEnter the element you want to search: ");

                scanf("%d", &tosearch);

            search(head, tosearch)? printf("\nYes, the element exists in the list.\n") :
    printf("\nNo, the element does not exist in the list");

        }

    return 0;

    }
```

Output screenshots:


C:\Users\arvin\OneDrive - morph B2B partnerships\Desktop\Desktop\lab3.exe

```
Enter your choice: 1. push() 2. quit insertion.
1
Enter the element: 10
Enter your choice: 1. push() 2. quit insertion.
1
Enter the element: 20
Enter your choice: 1. push() 2. quit insertion.
1
Enter the element: 30
Enter your choice: 1. push() 2. quit insertion.
2

Enter the element you want to search: 20

Yes, the element exists in the list.

--------------------------------
Process exited after 13.5 seconds with return value 0
Press any key to continue . . .
```

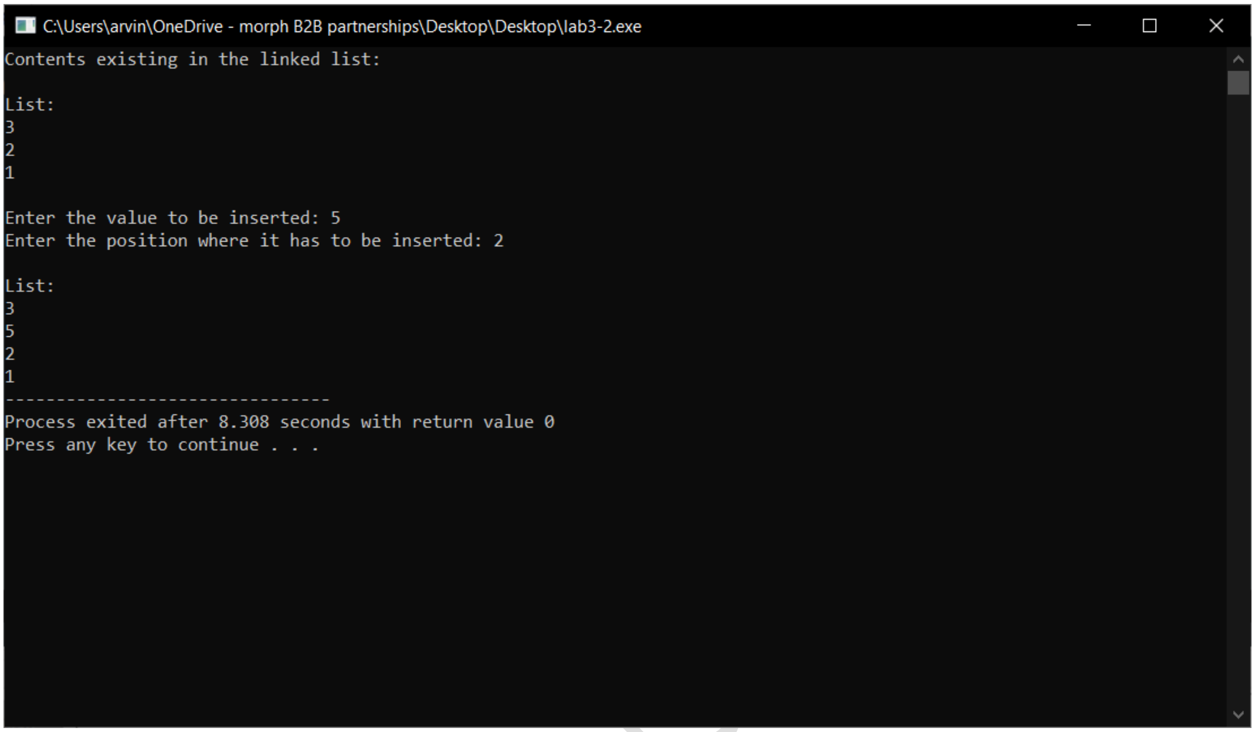

C:\Users\arvin\OneDrive - morph B2B partnerships\Desktop\Desktop\lab3.exe

```
Enter your choice: 1. push() 2. quit insertion.
1
Enter the element: 10
Enter your choice: 1. push() 2. quit insertion.
1
Enter the element: 20
Enter your choice: 1. push() 2. quit insertion.
1
Enter the element: 30
Enter your choice: 1. push() 2. quit insertion.
2

Enter the element you want to search: 40

No, the element does not exist in the list
--------------------------------
Process exited after 15.9 seconds with return value 0
Press any key to continue . . .
```

b. Insert after a specified value

```
#include<stdio.h>

#include<stdlib.h>

struct listNode *head=NULL;

struct listNode

{

    int data;

    struct listNode *next;

};

void insert(int data)

{

    struct listNode *temp = (struct listNode*)malloc(sizeof(struct listNode));

    temp->data=data;

    temp->next=head;
    head=temp;
}
void insert_at_position_n(int data,int position)
{
    struct listNode *ptr = (struct listNode*)malloc(sizeof(struct listNode));
    ptr->data=data;
    int i;
    struct listNode *temp=head;
    if(position==1)
    {
                    ptr->next=temp;
                    head=ptr;
                    return;
    }
            for(i=1;i<position-1;i++)
    {
                    temp=temp->next;
    }
    ptr->next=temp->next;
    temp->next=ptr;
}

void print()
{
    struct listNode *temp=head;
    printf("\nList:");
    while(temp!=NULL)
    {
                  printf("\n%d ",temp->data);
                    temp=temp->next;
       }
}

int main()
{
    insert(1);
    insert(2);
    insert(3);
    printf("Contents existing in the linked list: \n");
    print();
    printf("\n\nEnter the value to be inserted: ");
    int val;
    scanf("%d", &val);
    printf("Enter the position where it has to be inserted: ");
    int pos;
    scanf("%d", &pos);
       insert_at_position_n(val, pos);
    print();
    return 0;
}
```

Output:



C:\Users\arvin\OneDrive - morph B2B partnerships\Desktop\Desktop\lab3-2.exe

```
Contents existing in the linked list:

List:
3
2
1

Enter the value to be inserted: 5
Enter the position where it has to be inserted: 2

List:
3
5
2
1
--------------------------------
Process exited after 8.308 seconds with return value 0
Press any key to continue . . .
```

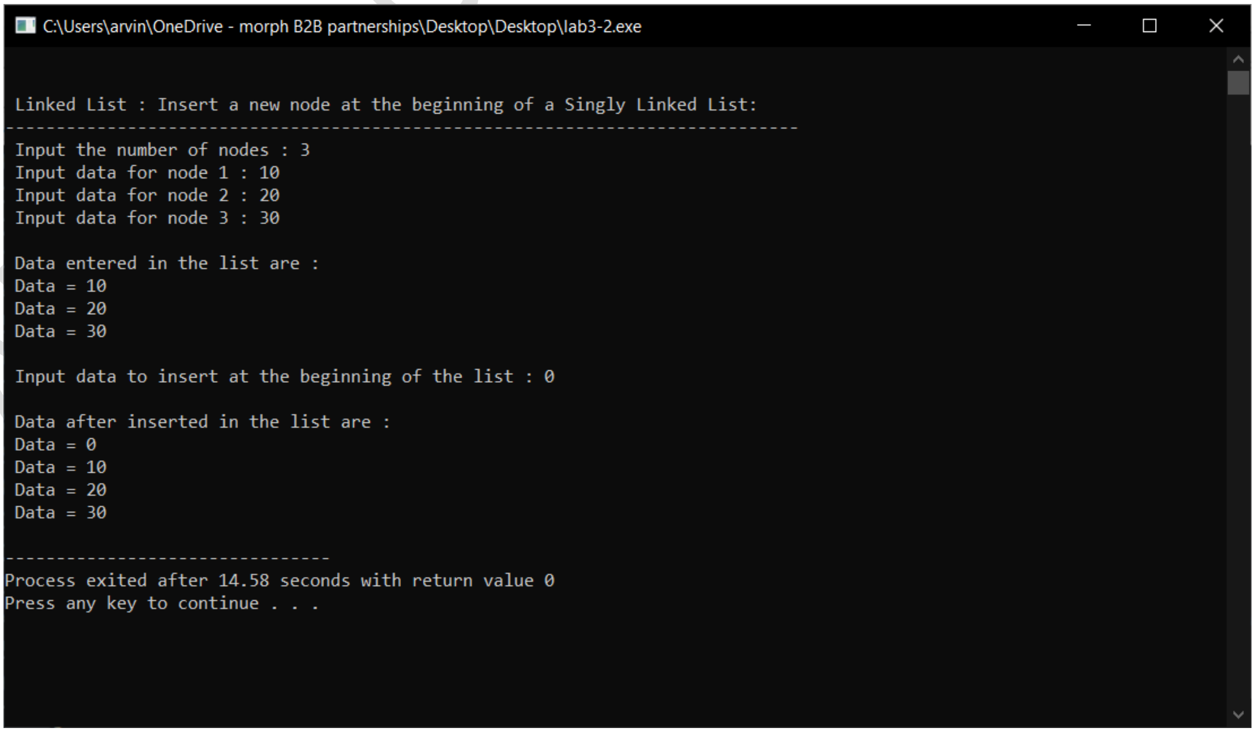
c. insert at beginning

Code:

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

struct node
{
    int num;                    //Data of the node
    struct node *nextptr;       //Address of the node
}*stnode;

void createNodeList(int n);     //function to create the list
void NodeInsertatBegin(int num);            //function to insert node at the
beginning
void displayList();             //function to display the list

int main()
{
    int n,num;
                    printf("\n\n Linked List : Insert a new node at the beginning of a
Singly Linked List:\n");
                    printf("----------------------------------------------------------------------
-----------\n");
    printf(" Input the number of nodes : ");
    scanf("%d", &n);
    createNodeList(n);
    printf("\n Data entered in the list are : \n");
    displayList();
    printf("\n Input data to insert at the beginning of the list : ");
    scanf("%d", &num);
    NodeInsertatBegin(num);
    printf("\n Data after inserted in the list are : \n");
    displayList();

    return 0;
}
void createNodeList(int n)
{
    struct node *fnNode, *tmp;
    int num, i;

    stnode = (struct node *)malloc(sizeof(struct node));
    if(stnode == NULL) //check whether the stnode is NULL and if so no memory
allocation
    {
        printf(" Memory can not be allocated.");
    }
    else
    {
// reads data for the node through keyboard
        printf(" Input data for node 1 : ");
        scanf("%d", &num);
        stnode-> num = num;
        stnode-> nextptr = NULL; //Links the address field to NULL
        tmp = stnode;
```

```
//Creates n nodes and adds to linked list
        for(i=2; i<=n; i++)
        {
            fnNode = (struct node *)malloc(sizeof(struct node));
            if(fnNode == NULL) //check whether the fnnode is NULL and if so no
memory allocation
            {
                printf(" Memory can not be allocated.");
                break;
            }
            else
            {
                printf(" Input data for node %d : ", i);
                scanf(" %d", &num);
                fnNode->num = num;      // links the num field of fnNode with num
                fnNode->nextptr = NULL; // links the address field of fnNode with
NULL
                tmp->nextptr = fnNode; // links previous node i.e. tmp to the fnNode
                tmp = tmp->nextptr;
            }
        }
    }
}
void NodeInsertatBegin(int num)
{
    struct node *fnNode;
    fnNode = (struct node*)malloc(sizeof(struct node));
    if(fnNode == NULL)
    {
        printf(" Memory can not be allocated.");
    }
    else
    {
        fnNode->num = num; //Links the data part
        fnNode->nextptr = stnode; //Links the address part
        stnode = fnNode; //Makes stnode as first node
    }
}

void displayList()
{
    struct node *tmp;
    if(stnode == NULL)
    {
        printf(" No data found in the list.");
    }
    else
    {
        tmp = stnode;
        while(tmp != NULL)
        {
            printf(" Data = %d\n", tmp->num);   // prints the data of current node
            tmp = tmp->nextptr;                 // advances the position of current node
        }
    }
}
```

Output:



C:\Users\arvin\OneDrive - morph B2B partnerships\Desktop\Desktop\lab3-2.exe

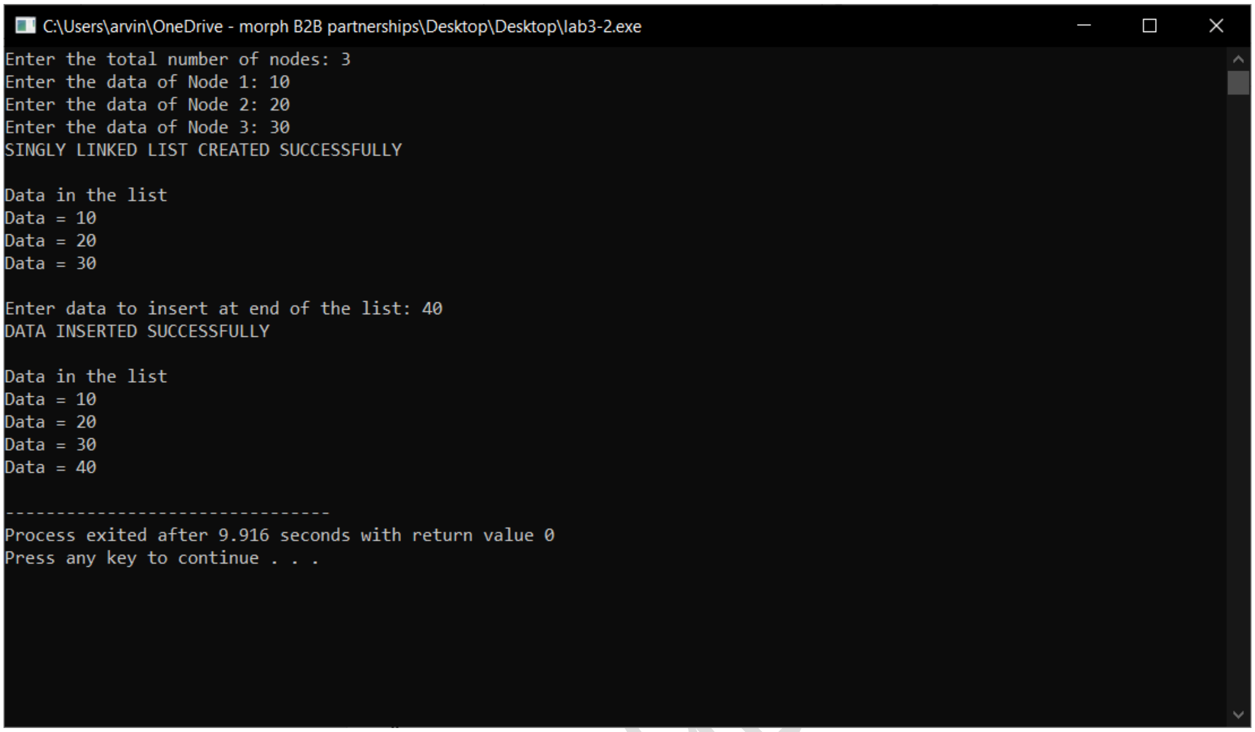
```


 Linked List : Insert a new node at the beginning of a Singly Linked List:
------------------------------------------------------------------------------
 Input the number of nodes : 3
 Input data for node 1 : 10
 Input data for node 2 : 20
 Input data for node 3 : 30

 Data entered in the list are :
 Data = 10
 Data = 20
 Data = 30

 Input data to insert at the beginning of the list : 0

 Data after inserted in the list are :
 Data = 0
 Data = 10
 Data = 20
 Data = 30

--------------------------------
Process exited after 14.58 seconds with return value 0
Press any key to continue . . .
```

d. insert at end

Code:

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


struct listNode {
    int key;
    struct listNode *next;
}*head;

void createList(int n)
{
    struct listNode *newNode, *temp;
    int key, i;
    head = (struct listNode *)malloc(sizeof(struct listNode));
    if(head == NULL)
    {
        printf("Unable to allocate memory.");
    }
    else
    {
        printf("Enter the data of Node 1: ");
        scanf("%d", &key);
        head->key = key;
        head->next = NULL;
        temp = head;
        for(i=2; i<=n; i++)
        {
            newNode = (struct listNode *)malloc(sizeof(struct listNode));
            if(newNode == NULL)
            {
                printf("Unable to allocate memory.");
                break;
            }
            else
            {
                printf("Enter the data of Node %d: ", i);
                scanf("%d", &key);
                newNode->key = key;
                newNode->next = NULL;
                temp->next = newNode;
                temp = temp->next;
            }
        }
        printf("SINGLY LINKED LIST CREATED SUCCESSFULLY\n");
    }
}
void insertNodeAtEnd(int key)
{
    struct listNode *newNode, *temp;
    newNode = (struct listNode*)malloc(sizeof(struct listNode));

    if(newNode == NULL)
    {
        printf("Unable to allocate memory.");
    }
    else
    {
        newNode->key = key;
        newNode->next = NULL;
        temp = head;
        while(temp->next != NULL)
            temp = temp->next;
        temp->next = newNode;
        printf("DATA INSERTED SUCCESSFULLY\n");
    }}
void displayList()
{
    struct listNode *temp;
    if(head == NULL)
    {
        printf("List is empty.");
    }
    else
    {
        temp = head;
        while(temp != NULL)

        {

            printf("Data = %d\n", temp->key);

            temp = temp->next;

        }}}

int main()

{

    int n, key;

    printf("Enter the total number of nodes: ");

    scanf("%d", &n);

    createList(n);

    printf("\nData in the list \n");

    displayList();

    printf("\nEnter data to insert at end of the list: ");

    scanf("%d", &key);

    insertNodeAtEnd(key);

    printf("\nData in the list \n");

    displayList();

    return 0;

}
```

Output:



C:\Users\arvin\OneDrive - morph B2B partnerships\Desktop\Desktop\lab3-2.exe

```
Enter the total number of nodes: 3
Enter the data of Node 1: 10
Enter the data of Node 2: 20
Enter the data of Node 3: 30
SINGLY LINKED LIST CREATED SUCCESSFULLY

Data in the list
Data = 10
Data = 20
Data = 30

Enter data to insert at end of the list: 40
DATA INSERTED SUCCESSFULLY

Data in the list
Data = 10
Data = 20
Data = 30
Data = 40

--------------------------------
Process exited after 9.916 seconds with return value 0
Press any key to continue . . .
```

e. Deletion of a particular element

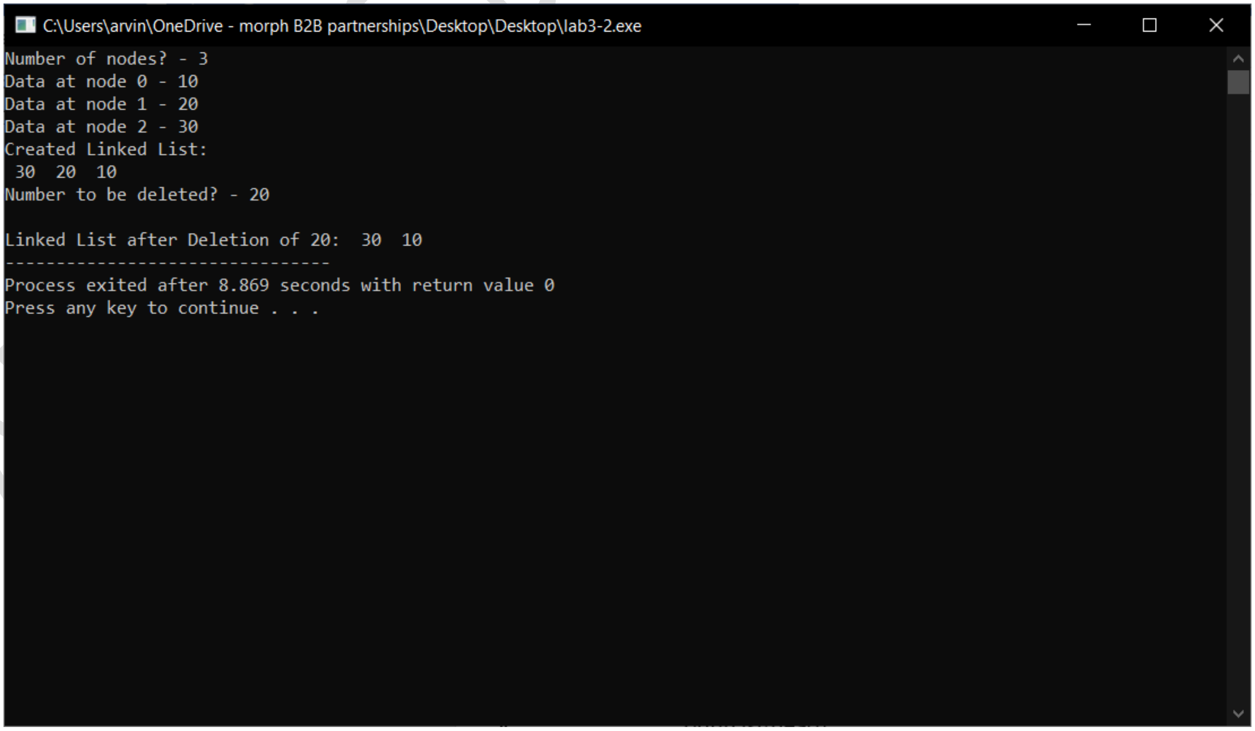
Code:

```
#include <stdio.h>
#include <stdlib.h>
struct Node
{
   int data;
   struct Node *next;
};
void push(struct Node** head_ref, int new_data)
{
   struct Node* new_node = (struct Node*) malloc(sizeof(struct Node));
   new_node->data  = new_data;
   new_node->next = (*head_ref);
   (*head_ref)    = new_node;
}
void deleteNode(struct Node **head_ref, int key)
```

```
{
    struct Node* temp = *head_ref, *prev;
    if (temp != NULL && temp->data == key)
    {
        *head_ref = temp->next;
        free(temp);
        return;
    }
    while (temp != NULL && temp->data != key)
    {
        prev = temp;
        temp = temp->next;
    }
    if (temp == NULL) return;
      prev->next = temp->next;

    free(temp);
}
void printList(struct Node *node)
{
    while (node != NULL)
    {
        printf(" %d ", node->data);
        node = node->next;
    }
}
int main()
{
    struct Node* head = NULL;
    printf("Number of nodes? - ");
    int no;
    scanf("%d", &no);
    for(int i=0; i<no; i++)
    {
           printf("Data at node %d - ", i);
           int data;
           scanf("%d", &data);
```

```
            push(&head, data);
            }
    puts("Created Linked List: ");
    printList(head);
    printf("\nNumber to be deleted? - ");
    int del;
    scanf("%d", &del);
    deleteNode(&head, del);
    printf("\nLinked List after Deletion of %d: ", del);
    printList(head);
    return 0;
}
```

Output:


C:\Users\arvin\OneDrive - morph B2B partnerships\Desktop\Desktop\lab3-2.exe

```
Number of nodes? - 3
Data at node 0 - 10
Data at node 1 - 20
Data at node 2 - 30
Created Linked List:
 30  20  10
Number to be deleted? - 20

Linked List after Deletion of 20:  30  10
--------------------------------
Process exited after 8.869 seconds with return value 0
Press any key to continue . . .
```

2. Write a program to represent a polynomial expression using linked list and implement functions to perform following operations. a. Polynomial addition b. Polynomial subtraction struct polyNode { int coeff; int pow; struct polyNode *next; }; Input: exp1: 4x^3+3x^2+2x+1 exp2: x^3+2x^2+3x+4 output: Addition: 5x^3+5x^2+5x+5 Subtraction: 3x^3+x^2-x-3

Code:

```
#include<stdio.h>

#include<stdlib.h>


struct listNode

{

    float coeff;

    int expo;

    struct listNode *link;

};


struct listNode *insert_s(struct listNode *start,float co,int ex)

{

    struct listNode *ptr,*tmp;

    tmp=(struct listNode *)malloc(sizeof(struct listNode));

    tmp->coeff=co;

    tmp->expo=ex;

    if(start==NULL || ex > start->expo)

    {

        tmp->link=start;

        start=tmp;
    }
    else
    {
        ptr=start;
        while(ptr->link!=NULL && ptr->link->expo >= ex)
            ptr=ptr->link;
        tmp->link=ptr->link;
        ptr->link=tmp;
    }
    return start;
}
struct listNode *create(struct listNode *start)
{
    int i,n,ex;
    float co;
    printf("Enter the number of terms : ");
    scanf("%d",&n);
    for(i=1;i<=n;i++)
    {
        printf("Enter coeficient for term %d : ",i);
        scanf("%f",&co);
        printf("Enter exponent for term %d : ",i);
        scanf("%d",&ex);
        start=insert_s(start,co,ex);
    }
    return start;
```

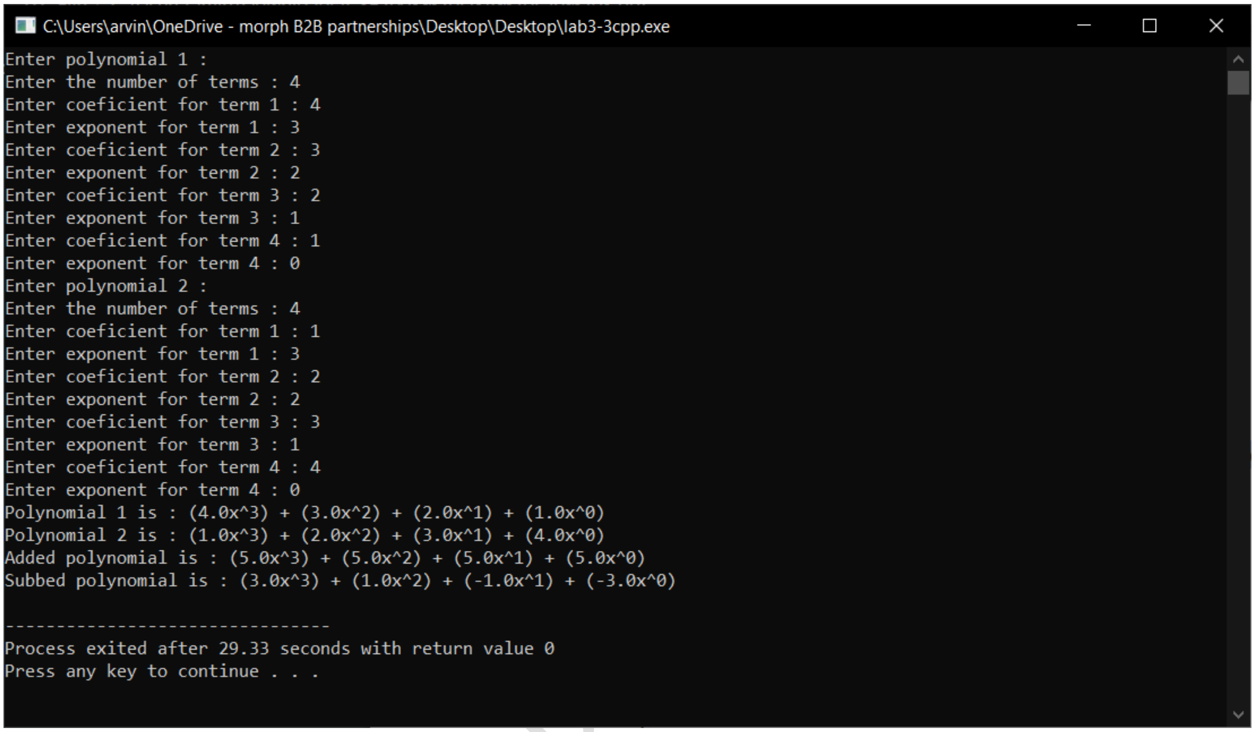
```
}



struct listNode *insert(struct listNode *start,float co,int ex)
{
    struct listNode *ptr,*tmp;
    tmp=(struct listNode *)malloc(sizeof(struct listNode));
    tmp->coeff=co;
    tmp->expo=ex;
    if(start==NULL)
    {
        tmp->link=start;
        start=tmp;
    }
    else
    {
        ptr=start;
        while(ptr->link!=NULL)
            ptr=ptr->link;
        tmp->link=ptr->link;
        ptr->link=tmp;
    }
    return start;
}

void display(struct listNode *ptr)
```

```
{
    if(ptr==NULL)
    {
        printf("Zero polynomial\n");
        return;
    }
    while(ptr!=NULL)
    {
        printf("(%.1fx^%d)", ptr->coeff,ptr->expo);
        ptr=ptr->link;
        if(ptr!=NULL)
            printf(" + ");
        else
            printf("\n");
    }
}

void poly_add(struct listNode *p1,struct listNode *p2)
{
    struct listNode *new_node;
    new_node=NULL;
    while(p1!=NULL && p2!=NULL)
    {
        if(p1->expo > p2->expo)
        {
            new_node=insert(new_node,p1->coeff,p1->expo);
            p1=p1->link;
        }
        else if(p2->expo > p1->expo)
        {
            new_node=insert(new_node,p2->coeff,p2->expo);
            p2=p2->link;
        }
        else if(p1->expo==p2->expo)
        {
            new_node=insert(new_node,p1->coeff+p2->coeff,p1->expo);
            p1=p1->link;
            p2=p2->link;
        }
    }
    while(p1!=NULL)
    {
        new_node=insert(new_node,p1->coeff,p1->expo);
        p1=p1->link;
    }
    while(p2!=NULL)
    {
        new_node=insert(new_node,p2->coeff,p2->expo);
        p2=p2->link;
    }
    printf("Added polynomial is : ");
    display(new_node);
```

```
}

void subtract(struct listNode *p1,struct listNode *p2)
{
    struct listNode *new_node;
    new_node=NULL;
    while(p1!=NULL && p2!=NULL)
    {
        if(p1->expo > p2->expo)
        {
            new_node=insert(new_node,p1->coeff,p1->expo);
            p1=p1->link;
        }
        else if(p2->expo > p1->expo)
        {
            new_node=insert(new_node,p2->coeff,p2->expo);
            p2=p2->link;
        }
        else if(p1->expo==p2->expo)
        {
            new_node=insert(new_node,p1->coeff- p2->coeff,p1->expo);
            p1=p1->link;
            p2=p2->link;
        }
    }
    while(p1!=NULL)
    {
        new_node=insert(new_node,p1->coeff,p1->expo);
        p1=p1->link;
    }
    while(p2!=NULL)
    {
        new_node=insert(new_node,p2->coeff,p2->expo);
        p2=p2->link;
    }
    printf("Subbed polynomial is : ");
    display(new_node);
}

int main( )
{
    struct listNode *start1=NULL,*start2=NULL;
    printf("Enter polynomial 1 :\n");
    start1=create(start1);
    printf("Enter polynomial 2 :\n");
    start2=create(start2);
    printf("Polynomial 1 is : ");
    display(start1);
    printf("Polynomial 2 is : ");
    display(start2);
    poly_add(start1, start2);
    subtract(start1, start2);
```

```
}
```

Output:


C:\Users\arvin\OneDrive - morph B2B partnerships\Desktop\Desktop\lab3-3cpp.exe

```
Enter polynomial 1 :
Enter the number of terms : 4
Enter coeficient for term 1 : 4
Enter exponent for term 1 : 3
Enter coeficient for term 2 : 3
Enter exponent for term 2 : 2
Enter coeficient for term 3 : 2
Enter exponent for term 3 : 1
Enter coeficient for term 4 : 1
Enter exponent for term 4 : 0
Enter polynomial 2 :
Enter the number of terms : 4
Enter coeficient for term 1 : 1
Enter exponent for term 1 : 3
Enter coeficient for term 2 : 2
Enter exponent for term 2 : 2
Enter coeficient for term 3 : 3
Enter exponent for term 3 : 1
Enter coeficient for term 4 : 4
Enter exponent for term 4 : 0
Polynomial 1 is : (4.0x^3) + (3.0x^2) + (2.0x^1) + (1.0x^0)
Polynomial 2 is : (1.0x^3) + (2.0x^2) + (3.0x^1) + (4.0x^0)
Added polynomial is : (5.0x^3) + (5.0x^2) + (5.0x^1) + (5.0x^0)
Subbed polynomial is : (3.0x^3) + (1.0x^2) + (-1.0x^1) + (-3.0x^0)

--------------------------------
Process exited after 29.33 seconds with return value 0
Press any key to continue . . .
```