Exercise 2: ListADT and its applications

Create an ADT for the linked list data structure with the following functions. list*ADT* will have the integer array and size.

a. insert(header,data) – Insert data into the list using inserting at front

b. display(header) – Display the elements of the list

c. insertAtEnd(header,data) – Insert data at the end of the list

d. searchElt(header, key) – return the value if found, otherwise return -1

e. findMiddleElt(header) – find the middle element in the list

f. reverseList(header) – Reverse the list

g. length(header) – find the length of the list

h. deleteElt(header,data) – Deletes the element data

Write a program in C to test the listADT for its operations with the following test cases.

Operation Expected Output

length(header) 0

insert(header,2) 2

insert(header,4) 4, 2

insert(header,6) 6, 4, 2

insert(header,8) 8, 6, 4, 2

length(header) 4

insertLast(header,1) 8, 6, 4, 2, 1

insertLast(header,3) 8, 6, 4, 2, 1, 3

length(header) 6

findMiddleElt(header) 2 or 4

reverseList(header) 3, 1, 2, 4, 6, 8

searchElt(4) 4

searchElt(5) -1

deleteElt(2) 8, 6, 4, 1, 3

Best practices to be followed:

• Design before coding

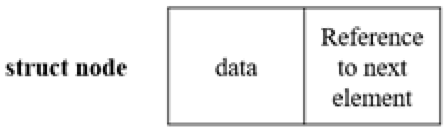
• Usage of algorithm notation

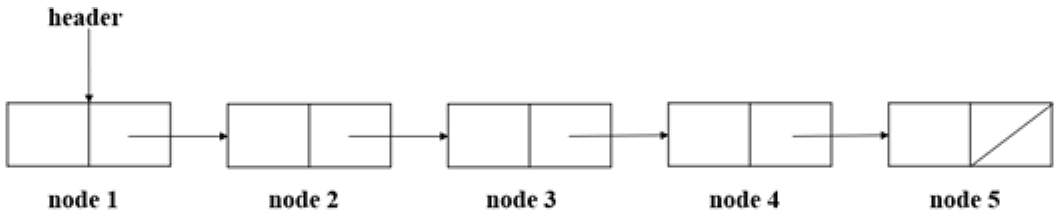
• Use of multi-file C program

• Versioning of code

Write a program to subtract the given two polynomials

DESIGN:





CODE:

main.c

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

void main()
{
        struct node*header;
```

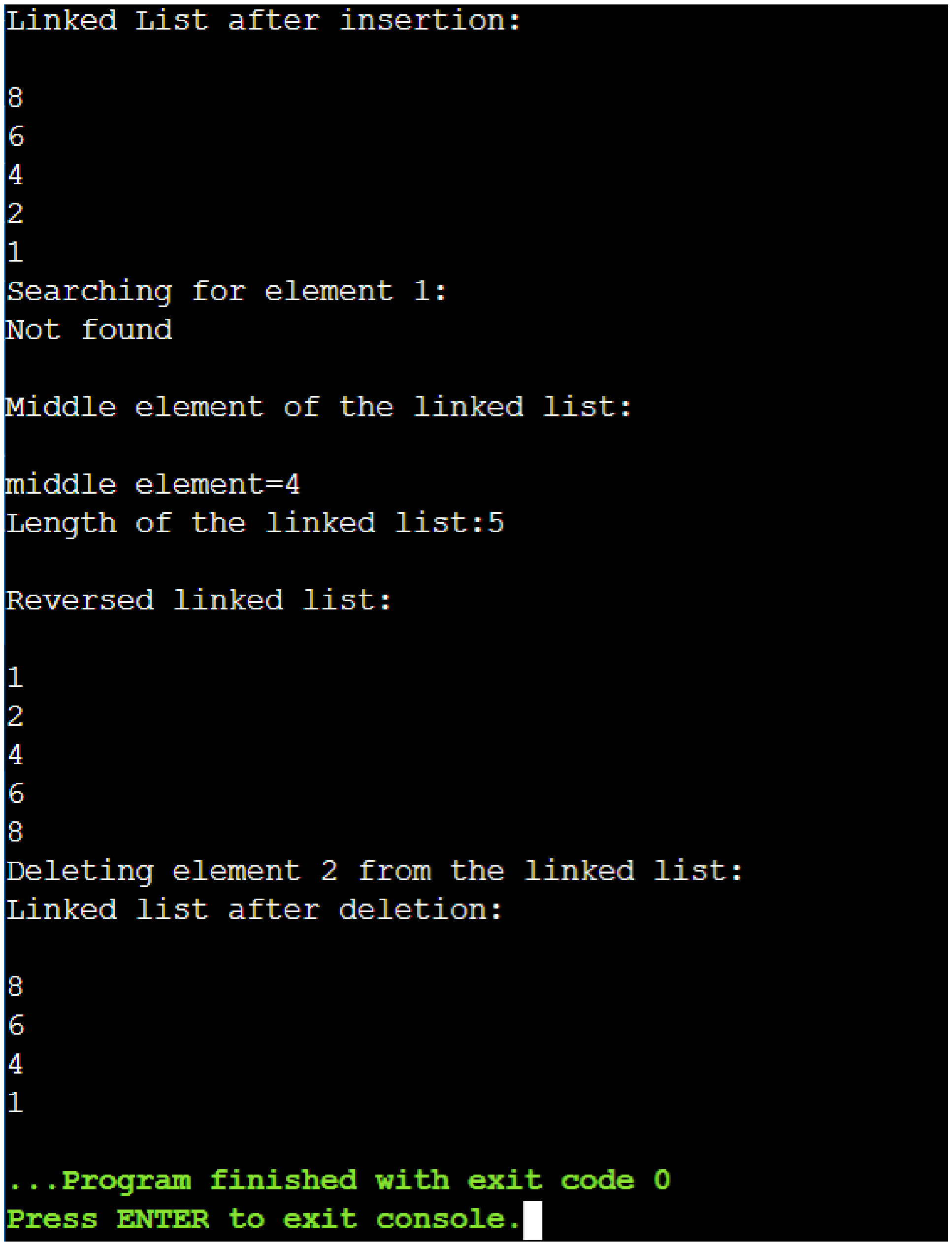
```
            struct node *l;
            l=(struct node *)malloc(sizeof(struct node));
           header=(struct node*)malloc(sizeof(struct node));
           header->next=NULL;
           int m;
           insert(header,2);
           insert(header,4);
           insert(header,6);
           insert(header,8);

           insertAtEnd(header,1);
           m=search(header,1);
           if(m==1)
           {
           printf("\nvalue is found ");
           }
   else{
           printf("\nnot found");
   }
   findMiddleElt(header);
   lengh(header);
    l=reverse(p);
   display(l);
   deletenode(header,2);
   display(header);
 }
```

arrADT.h

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

};
void insert(struct node*header,int Data){
          struct node *temp;
```

```
        temp=(struct node*)malloc(sizeof(struct node));
        temp->data=Data;
  temp->next= header->next;
  header->next=temp;

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

        }

}

void insertAtEnd(struct node*header,int data)
{
   struct node*temp;
   temp = (struct node*)malloc(sizeof(struct node));
   temp->data = data;
   temp->next = NULL;

   if (header->next == NULL) {
      header->next = temp;
      return;
   }

   struct node*ptr = header->next;
   while (ptr->next != NULL) {
      ptr = ptr->next;
   }
   ptr->next = temp;
}


int search(struct node*header,int key){
  struct node*ptr;
  ptr=header->next;
  while(ptr!=NULL){
  if(ptr->data==key){
          return 1;
  }
          ptr=ptr->next;
  }

return -1;


}
int lengh(struct node*header){
  struct node*ptr;
  int count=0;
  ptr=header->next;
  while(ptr!=NULL)
          {
          count++;
          ptr=ptr->next;
          }
  return count;
 // printf("\nlengh of list is=%d",count);

}
void findMiddleElt(struct node*header){
  struct node*ptr;
 int count=0;
  int m;
  ptr=header->next;
  m=lengh(header);
  int n=m/2;

  while(ptr!=NULL)
          {

          if(count==n)
          {
          printf("\nmiddle element=%d",ptr->data);
          break;
          }
          else {
          count++;
          ptr=ptr->next;
          }

          }
}
struct node* reverse(struct node *h)
{
   int a[100];
   int i = 0;

   if (h == NULL)
      return NULL;

   struct node* ptr = h->next;
   while (ptr != NULL) {
      a[i++] = ptr->data;
      ptr = ptr->next;
   }

   struct node *h1;
   h1 = (struct node*)malloc(sizeof(struct node));
   h1->next = NULL;   // ✅ CRITICAL FIX

   for (int j = i - 1; j >= 0; j--) {
      insertAtEnd(h1, a[j]);
   }

   return h1;
}

void deletenode(struct node *header,int key)
{
  struct node *prev;
  struct node*ptr;
  prev=header;
  ptr=header->next;
  while(ptr!=NULL)
        {
        if(ptr->data==key)
        {
        prev->next=ptr->next;
        free(ptr);
        break;
        }
        else
        {
        prev=ptr;
        ptr=ptr->next;
        }
        }
}
```

Output:



```
Linked List after insertion:

8
6
4
2
1
Searching for element 1:
Not found

Middle element of the linked list:

middle element=4
Length of the linked list:5

Reversed linked list:

1
2
4
6
8
Deleting element 2 from the linked list:
Linked list after deletion:

8
6
4
1

...Program finished with exit code 0
Press ENTER to exit console.
```

LEARNING OUTCOME;

- It lays the foundation for understanding essential data structure.
- And,it cultivates algorithmic thinking through the implementation of key operations such as insertion,deletion and searching.
- I acquire proficiency in data manipulation techniques, optimizing the efficiency of list operations.
- I gain insights into code optimization and reduce time complexity.