Date: 19.09.2023 Name: Niranjan.B
Ex. No: 2 Reg. No.: 3122225001082

UCS 2312 Data Structures Lab
Exercise 2: ListADT and its applications

Date of Exercise: 19.09.2023

Create an ADT for the linked list data structure with the following functions. list*ADT* will have the integer array and size. [CO1, K3]

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. detectLoop(header) – return the status of whether the loop is present or not
h. deleteElt(header,data) – Deletes the element data

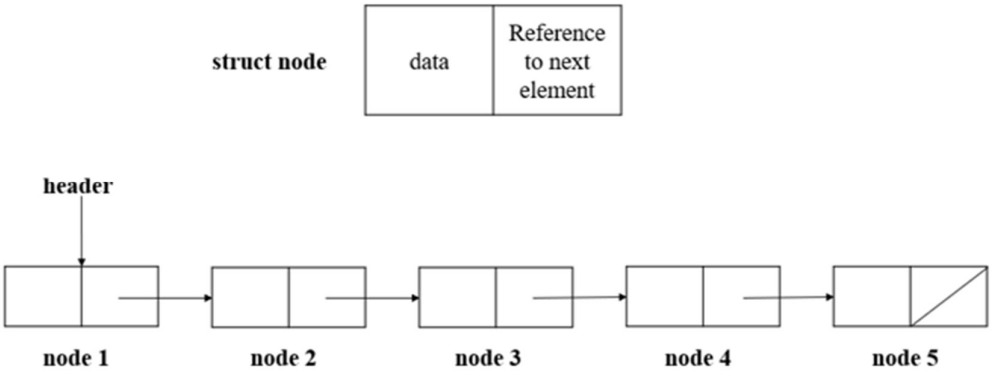
Write a program in C to test the listADT for its operations with the following test cases.

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

Data Structure – Linked List:



Algorithm –

Algorithm: Find the middle element in the list

Input – Pointer to header node
Output – void

1. count=0 and mid=header
2. while (header != NULL)
   if count & 1
   mid=mid->next
   ++count
   header=header->next
3. if mid != NULL
   print mid->data

Algorithm: Reverse the list

Input – Pointer to header node
Output – void

1. prev=NULL
2. curr=header->next
3. while (curr != NULL)
   next = curr->next
   curr->next = prev
   prev = curr
   curr = next
4. header->next = prev

Algorithm: Return the status of whether the loop is present or not

Input – Pointer to header node
Output – int

1. ptr1 = header->next
2. ptr2 = header
3. while (ptr1 != NULL)
   while (ptr2->next != ptr1)
   if ptr2 == ptr1->next
   return 1
   ptr2 = ptr2->next
   ptr2 = header
   ptr1 = ptr1->next
4. return 0

main.c code:

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

void main()
{

      struct node *head = (struct node*) malloc(sizeof(struct node));
      head->next = NULL;
      printf("ENTER FIRST ELEMENT: ");
      int value;
      scanf("%d", &value);
      create(head, value);

      int choice;
      while(choice + 1)
      {
            printf("\n-1: EXIT\n 0: DISPLAY\n 1: APPEND(INSERT AT END)\n
2: INSERT AT BEGINNING\n 3: INSERT AFTER DATA\n 4: DELETE AT FRONT");
            printf("\n 5: DELETE AT END\n 6: DELETE DATA\n 7: SEARCH
DATA\n 8: REVERSE\n 9: MIDDLE ELEMENT\n 10: DETECT LOOP\nChoice : ");
            scanf("%d", &choice);
            switch (choice)
            {
                  case -1: break;
                  case 0:
                  {
                        display(head);
                        break;
                  }
                  case 1:
                  {
                        printf("\nENTER NEW ELEMENT: ");
                        int data;
                        scanf("%d", &data);
                        append(head, data);
                        printf("New List : ");
                        display(head);
```

```
            break;
        }
        case 2:
        {
            printf("\nENTER NEW ELEMENT: ");
            int data;
            scanf("%d", &data);
            insertAtFront(head, data);
            printf("New List : ");
            display(head);
            break;
        }
        case 3:
        {
            printf("\nENTER NEW ELEMENT: ");
            int data;
            scanf("%d", &data);

            printf("\nENTER KEY: ");
            int key;
            scanf("%d", &key);

            insertAfter(head, data, key);
            printf("New List : ");
            display(head);
            break;
        }
        case 4:
        {
            deleteAtFront(head);
            printf("New List : ");
            display(head);
            break;
        }
        case 5:
        {
            deleteAtEnd(head);
            printf("New List : ");
            display(head);
            break;
        }
        case 6:
        {
            printf("\nENTER ELEMENT TO DELETE: ");
            int data;
            scanf("%d", &data);
            deleteData(head, data);
            printf("New List : ");
            display(head);
            break;

        }
        case 7:
        {
```

```
                        printf("\nENTER DATA TO SEARCH: ");
                        int data;
                        scanf("%d", &data);
                        if(search(head, data)!=NULL)
                        {
                              printf("ELEMENT FOUND\n");
                        }
                        else
                        {
                              printf("ELEMENT NOT FOUND\n");
                        }
                        break;

                  }
                  case 8:
                  {
                        printf("\nLIST REVERSED ");
                        reverse(head);
                        printf("New List : ");
                        display(head);
                        break;
                  }
                  case 9:
                  {
                        printMiddle(head);
                  }
                  case 10:
                  {
                        if(detectLoop(head))
                        {
                              printf("\nLoop Present\n");
                        }
                        else
                        {
                              printf("\nNo Loop Present\n");
                        }
                        break;
                  }
                  case 11:
                  {
                        head->next->next->next->next=head->next;
                        break;
                  }
                  default:
                  {
                        printf("\nINVALID CHOICE");
                  }

            }
      }
}
```

LinkedADT.h code:

```
struct node
{
      int data;
      struct node * next;
};

void create(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 append(struct node* header,int data)
{
      struct node* temp;
      temp=(struct node *)malloc(sizeof(struct node));
      temp->data=data;
      struct node* ptr;
      ptr=header->next;
      while((ptr->next)!=NULL)
      {
            ptr=ptr->next;
      }
      temp->next=ptr->next;
      ptr->next=temp;
}

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

void insertAtFront(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 insertAfter(struct node* header,int data,int key)
{
      struct node* temp;
      temp=(struct node *)malloc(sizeof(struct node));
      temp->data=data;
      struct node* ptr;
      ptr=search(header,key);
      temp->next=ptr->next;
      ptr->next=temp;
}

void deleteData(struct node* header,int data)
{
      struct node* ptr;
      ptr=header->next;
      while(ptr!=NULL)
      {
            if((ptr->next->data)==data)
            {
                  ptr->next=ptr->next->next;
            }
            ptr=ptr->next;
      }
}

void deleteAtFront(struct node *header)
{
      struct node *temp = header->next;
      header->next = temp->next;
      free(temp);
}

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

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

```
      printf("\n");
}

void reverse(struct node * header)
{
      struct node* prev=NULL;
    struct node* curr=header->next;
    struct node* next=NULL;
    while (curr != NULL)
      {
         next = curr->next;
         curr->next = prev;
         prev = curr;
         curr = next;
    }
    header->next = prev;
}

void printMiddle(struct node* header)
{
    int count=0;
    struct node* mid=header;
    while(header!=NULL)
    {
        if(count & 1)
            mid=mid->next;
        ++count;
        header=header->next;
    }
    if (mid!=NULL)
        printf("The middle element is %d\n",mid->data);
}

int detectLoop(struct node* header)
{
      struct node *ptr1=header->next;
      struct node *ptr2=header;
      while(ptr1!=NULL)
      {
            while(ptr2->next!=ptr1)
            {
                  if(ptr2==ptr1->next)
                  {
                        return 1;
                  }
                  ptr2=ptr2->next;
            }
            ptr2=header;
            ptr1=ptr1->next;
      }
      return 0;
}
```

Output:

```
PS D:\College\Sem 3\Data Structures\Assignment 2> gcc main.c
PS D:\College\Sem 3\Data Structures\Assignment 2> ./a.exe
ENTER FIRST ELEMENT: 1

-1: EXIT
 0: DISPLAY
 1: APPEND(INSERT AT END)
 2: INSERT AT BEGINNING
 3: INSERT AFTER DATA
 4: DELETE AT FRONT
 5: DELETE AT END
 6: DELETE DATA
 7: SEARCH DATA
 8: REVERSE
 9: MIDDLE ELEMENT
 10: DETECT LOOP
Choice : 2

ENTER NEW ELEMENT: 4
New List : Linked List => 4 1

-1: EXIT
 0: DISPLAY
 1: APPEND(INSERT AT END)
 2: INSERT AT BEGINNING
 3: INSERT AFTER DATA
 4: DELETE AT FRONT
 5: DELETE AT END
 6: DELETE DATA
 7: SEARCH DATA
 8: REVERSE
 9: MIDDLE ELEMENT
 10: DETECT LOOP
Choice : 0
Linked List => 4 1
```

Name: Niranjan.B
Reg. No.: 3122225001082

```
-1: EXIT
 0: DISPLAY
 1: APPEND(INSERT AT END)
 2: INSERT AT BEGINNING
 3: INSERT AFTER DATA
 4: DELETE AT FRONT
 5: DELETE AT END
 6: DELETE DATA
 7: SEARCH DATA
 8: REVERSE
 9: MIDDLE ELEMENT
 10: DETECT LOOP
Choice : 1

ENTER NEW ELEMENT: 6
New List : Linked List => 4 1 6

-1: EXIT
 0: DISPLAY
 1: APPEND(INSERT AT END)
 2: INSERT AT BEGINNING
 3: INSERT AFTER DATA
 4: DELETE AT FRONT
 5: DELETE AT END
 6: DELETE DATA
 7: SEARCH DATA
 8: REVERSE
 9: MIDDLE ELEMENT
 10: DETECT LOOP
Choice : 7

ENTER DATA TO SEARCH: 1
ELEMENT FOUND
```

```
-1: EXIT
 0: DISPLAY
 1: APPEND(INSERT AT END)
 2: INSERT AT BEGINNING
 3: INSERT AFTER DATA
 4: DELETE AT FRONT
 5: DELETE AT END
 6: DELETE DATA
 7: SEARCH DATA
 8: REVERSE
 9: MIDDLE ELEMENT
 10: DETECT LOOP
Choice : 9
The middle element is 1
```

Name: Niranjan.B
Reg. No.: 3122225001082

```
-1: EXIT
 0: DISPLAY
 1: APPEND(INSERT AT END)
 2: INSERT AT BEGINNING
 3: INSERT AFTER DATA
 4: DELETE AT FRONT
 5: DELETE AT END
 6: DELETE DATA
 7: SEARCH DATA
 8: REVERSE
 9: MIDDLE ELEMENT
 10: DETECT LOOP
Choice : 10

No Loop Present

-1: EXIT
 0: DISPLAY
 1: APPEND(INSERT AT END)
 2: INSERT AT BEGINNING
 3: INSERT AFTER DATA
 4: DELETE AT FRONT
 5: DELETE AT END
 6: DELETE DATA
 7: SEARCH DATA
 8: REVERSE
 9: MIDDLE ELEMENT
 10: DETECT LOOP
Choice : 8

LIST REVERSED New List : Linked List => 6 1 4
```

POLYNOMIAL ADDITION:

Algorithm: Add two polynomials

1. t=sum
2. while (p1->next != NULL)
   insert (sum, p1->next->coef, p1->next->exp)
   p1 = p1->next
3. while (p2->next != NULL)
   flag=0
   sum=t
   while (sum->next != NULL)
   if (p2->next->exp == sum->next->exp)
   flag=1
   sum->next->coef = sum->next->coef + p2->next->coef
   sum = sum->next
   if flag==0
   insert (sum, p2->next->coef, p2->next->exp)
   p2 = p2->next

Polynomial.h code:

```
struct node
{
    int coef;
    int exp;
    struct node *next;
};

void insert(struct node *header, int coef, int exp)
{
    struct node *temp=(struct node *)malloc(sizeof(struct node));
    temp->coef=coef;
    temp->exp=exp;
    temp->next=NULL;
    if(header->next==NULL)
    {
        header->next=temp;
    }
    else
    {
        struct node *current=header->next;
        while (current->next!=NULL)
        {
            current=current->next;
        }
        current->next=temp;
    }
}

void display(struct node *p)
{
    struct node *current=p->next;
    while(current!=NULL)
    {
        printf("%dx^%d",current->coef,current->exp);
```

```
        current=current->next;
        if (current!=NULL)
        {
            printf(" + ");
        }
    }
    printf("\n");
}

void add(struct node* p1, struct node* p2, struct node* sum)
{
    int flag;
    struct node* t=sum;
    while (p1->next!=NULL)
    {
        insert(sum,p1->next->coef,p1->next->exp);
        p1=p1->next;
    }
    while (p2->next!=NULL)
    {
        flag=0;
        sum=t;
        while (sum->next!=NULL)
        {
            if(p2->next->exp==sum->next->exp)
            {
                flag=1;
                sum->next->coef=sum->next->coef+p2->next->coef;
            }
            sum=sum->next;
        }
        if(flag==0)
        {
            insert(sum,p2->next->coef,p2->next->exp);
        }
        p2=p2->next;
    }
}
```

PolynomialMain.c code:

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

void main()
{
    struct node *p1=(struct node *)malloc(sizeof(struct node));
    struct node *p2=(struct node *)malloc(sizeof(struct node));
    struct node *sum=(struct node *)malloc(sizeof(struct node));
    p1->next=NULL;
    p2->next=NULL;
    sum->next=NULL;
    int choice;
     while(choice + 1)
```

```
      {
            printf("\n-1: EXIT\n 0: DISPLAY POLYNOMIALS\n 1: INSERT AT
POLYNOMIAL 1\n 2: INSERT AT POLYNOMIAL 2\n 3: DISPLAY SUM\nChoice : ");
            scanf("%d", &choice);
            switch (choice)
            {
                  case -1: break;
                  case 0:
                  {
                        printf("Polynomial 1: ");
                              display(p1);
                              printf("Polynomial 2: ");
                              display(p2);
                        break;
                  }
                  case 1:
                  {
                        printf("\nENTER COEFFICIENT AND EXPONENT: \n");
                        int coef,exp;
                        scanf("%d",&coef);
                              scanf("%d",&exp);
                        insert(p1,coef,exp);
                        break;
                  }
                                          case 2:
                  {
                        printf("\nENTER COEFFICIENT AND EXPONENT: \n");
                        int coef,exp;
                        scanf("%d",&coef);
                              scanf("%d",&exp);
                        insert(p2,coef,exp);
                        break;
                  }
                              case 3:
                  {
                        printf("\nPOLYNOMIALS ADDED\n");
                        add(p1,p2,sum);
                              printf("Sum: ");
                              display(sum);
                        break;
                  }
                  default:
                  {
                        printf("\nINVALID CHOICE");
                  }

            }
      }
}
```

Output:

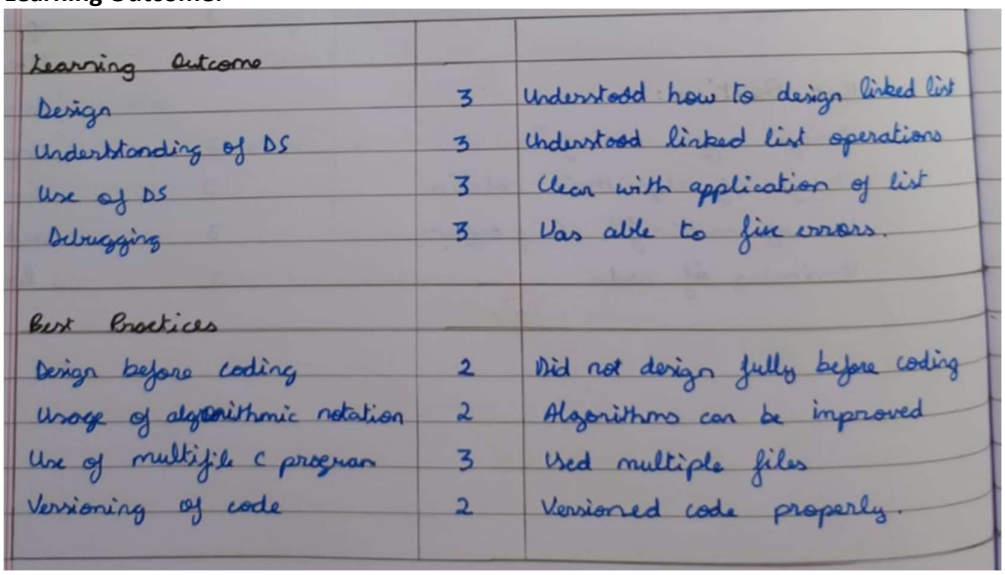
```
PS D:\College\Sem 3\Data Structures\Assignment 2> gcc PolynomialMain.c
PS D:\College\Sem 3\Data Structures\Assignment 2> ./a.exe

-1: EXIT
 0: DISPLAY POLYNOMIALS
 1: INSERT AT POLYNOMIAL 1
 2: INSERT AT POLYNOMIAL 2
 3: DISPLAY SUM
Choice : 1

ENTER COEFFICIENT AND EXPONENT:
2
3

-1: EXIT
 0: DISPLAY POLYNOMIALS
 1: INSERT AT POLYNOMIAL 1
 2: INSERT AT POLYNOMIAL 2
 3: DISPLAY SUM
Choice : 1

ENTER COEFFICIENT AND EXPONENT:
4
2

-1: EXIT
 0: DISPLAY POLYNOMIALS
 1: INSERT AT POLYNOMIAL 1
 2: INSERT AT POLYNOMIAL 2
 3: DISPLAY SUM
Choice : 1

ENTER COEFFICIENT AND EXPONENT:
6
1
```

Name: Niranjan.B
Reg. No.: 3122225001082

```
-1: EXIT
 0: DISPLAY POLYNOMIALS
 1: INSERT AT POLYNOMIAL 1
 2: INSERT AT POLYNOMIAL 2
 3: DISPLAY SUM
Choice : 2

ENTER COEFFICIENT AND EXPONENT:
3
3

-1: EXIT
 0: DISPLAY POLYNOMIALS
 1: INSERT AT POLYNOMIAL 1
 2: INSERT AT POLYNOMIAL 2
 3: DISPLAY SUM
Choice : 2

ENTER COEFFICIENT AND EXPONENT:
5
1

-1: EXIT
 0: DISPLAY POLYNOMIALS
 1: INSERT AT POLYNOMIAL 1
 2: INSERT AT POLYNOMIAL 2
 3: DISPLAY SUM
Choice : 0
Polynomial 1: 2x^3 + 4x^2 + 6x^1
Polynomial 2: 3x^3 + 5x^1

-1: EXIT
 0: DISPLAY POLYNOMIALS
 1: INSERT AT POLYNOMIAL 1
 2: INSERT AT POLYNOMIAL 2
 3: DISPLAY SUM
Choice : 3

POLYNOMIALS ADDED
Sum: 5x^3 + 4x^2 + 11x^1
```

Date: 19.09.2023 Name: Niranjan.B
Ex. No: 2 Reg. No.: 3122225001082

Learning Outcome:



Learning Outcome		
Design	3	Understood how to design linked list
Understanding of DS	3	Understood linked list operations
Use of DS	3	Clear with application of list
Debugging	3	Was able to fix errors.
Best Practices		
Design before coding	2	Did not design fully before coding
Usage of algorithmic notation	2	Algorithms can be improved
Use of multifile C program	3	Used multiple files
Versioning of code	2	Versioned code properly.