*NAME - AKRITI CHOUDHARY*

*ROLL NUMBER - 2005776*

*SUBJECT - DSA LAB*

*DATE - 7/9/2021*

*CLASS - B14*

*BRANCH - CSE*

## *Question 1 :WAP to create a linked list that represents a polynomial expression with single variable (i.e.5x7-3x5+x2+9) and display the polynomial by using user defined functions for creation and display.*

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

/*Traversal of the list*/
void traverse(struct node *h)
{
   if (h == NULL)
   {
      printf("\nThe list is empty\n");
   }
   else
   {
      while (h != NULL)
      {
         if (h->data >= 0)
            printf("+ %dx^%d  ", h->data, h->p);
         else
            printf("%dx^%d  ", h->data, h->p);

         h = h->next;
      }
   }
   puts("");
}

/*To check whether the list is empty or not*/
void isEmpty(struct node *h)
{
   if (h == NULL)
   {
      printf("\n The list is empty\n");
   }
   else
   {
      printf("\n The list is not empty\n");
   }
   puts("");
}

/*To insert a node at a given position in the list*/
void insert(struct node **h, int pos)
{
   struct node *cur = NULL;
   cur = (struct node *)malloc(sizeof(struct node));
   if (cur == NULL)
```

```
    {
        puts("Memory is not allocated");
    }
    printf("\nEnter the coefficient for the created node \n");
    scanf("%d", &cur->data);
    printf("\nEnter the power for the created node \n");
    scanf("%d", &cur->p);
    cur->next = NULL;

    if (*h == NULL) //to insert a node in an empty list
    {
        *h = cur;
    }
    else if (pos == 0) //to insert a node at the beginning of the list
    {
        cur->next = *h;
        *h = cur;
    }
    else
    {
        struct node *tmp = *h;
        int i = 0;
        while ((i < pos - 1) && (tmp->next != NULL))
        {
            tmp = tmp->next;
            ++i;
        }
        cur->next = tmp->next;
        tmp->next = cur;
    }
    puts("");
}

/*To delete a node at a given position from the list*/
void deleteNode(struct node **h, int pos)
{
    if (*h == NULL)
    {
        puts("The list is empty");
    }
    else if ((pos == 0) && ((*h)->next == NULL))
    {
        free(*h);
        *h = NULL;
    }

    else
    {
        struct node *tmp = *h, *cur = *h;
        int i = 0;
        while ((i <= pos - 1) && (cur->next != NULL))
        {
            ++i;
            tmp = cur;
            cur = cur->next;
        }
        if (pos == 0)
        {
```

```
            *h = cur->next;
            free(cur);
        }
        else if ((*h)->next == NULL)
        {
            free(*h);
            *h = NULL;
        }
        else
        {
            tmp->next = cur->next;
            free(cur);
        }
    }
    puts("");
}

int main()
{
    struct node *head = NULL;
    int position;
    int ch = 0;
    while (ch != -1)
    {
        puts("Enter the choice :");
        puts("1 - Traversal of the list\n"
             "2 - Check if the list is empty\n"
             "3 - Insert a node at the certain position (at beginning/end/any position)\n"
             "4 - Delete a node at the certain position (at beginning/end/any position)\n"
             "-1 - To exit");
        puts("");

        scanf("%d", &ch);
        switch (ch)
        {
        case 1:
            traverse(head);
            break;

        case 2:
            isEmpty(head);
            break;

        case 3:
            puts("Enter the position at which the node is to be inserted");
            scanf("%d", &position);
            insert(&head, position);
            break;
        case 4:
            puts("Enter the position at which the node is to be deleted");
            scanf("%d", &position);
            deleteNode(&head, position);
            break;
        case -1:
            puts("-----------------------Terminated----------------------------");
            break;
        default:
            puts("Wrong choice");
```

```
        break;
      }
    }
    return 0;
}
```

```
PS D:\KIIT_NOTES\2nd year sem_3\dsa_lab\7_9_2021> ./polyDisplay
Enter the choice :
1 - Traversal of the list
2 - Check if the list is empty
3 - Insert a node at the certain position (at beginning/end/any position)
4 - Delete a node at the certain position (at beginning/end/any position)
-1 - To exit

3
Enter the position at which the node is to be inserted
0

Enter the coefficient for the created node
-2

Enter the power for the created node
2

Enter the choice :
1 - Traversal of the list
2 - Check if the list is empty
3 - Insert a node at the certain position (at beginning/end/any position)
4 - Delete a node at the certain position (at beginning/end/any position)
-1 - To exit

3
Enter the position at which the node is to be inserted
4

Enter the coefficient for the created node
34

Enter the power for the created node
102

Enter the choice :
1 - Traversal of the list
2 - Check if the list is empty
3 - Insert a node at the certain position (at beginning/end/any position)
4 - Delete a node at the certain position (at beginning/end/any position)
-1 - To exit

1
-2x^2  + 34x^102
Enter the choice :
```

```
Enter the choice :
1 - Traversal of the list
2 - Check if the list is empty
3 - Insert a node at the certain position (at beginning/end/any position)
4 - Delete a node at the certain position (at beginning/end/any position)
-1 - To exit

2

 The list is empty

Enter the choice :
1 - Traversal of the list
2 - Check if the list is empty
3 - Insert a node at the certain position (at beginning/end/any position)
4 - Delete a node at the certain position (at beginning/end/any position)
-1 - To exit

-1
-------------------------Terminated-----------------------------
PS D:\KIIT_NOTES\2nd year sem_3\dsa_lab\7_9_2021>
```

## *Question2 : WAP to add two polynomials with single variable.*

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
   int exp;
   int cof;
   struct node *next;
};
void create(struct node **h)
{
   int n;
   struct node *cur, *ptr;
   printf("enter the no. of terms in the polynomial");
   scanf("%d", &n);
   for (int i = 0; i < n; i++)
   {
      cur = (struct node *)malloc(sizeof(struct node));
      printf("enter the coefficient and the exponent of the %dth term", i + 1);
      scanf("%d%d", &cur->cof, &cur->exp);
      cur->next = NULL;
      if (*h == NULL)
      {
         *h = cur;
         ptr = cur;
      }
      else
      {
         ptr->next = cur;
         ptr = cur;
```

```
        }
    }
}
void join(struct node **h1, struct node *h2)
{
    if (*h1 == NULL)
        *h1 = h2;
    else
    {
        struct node *ptr;
        for (ptr = *h1; ptr->next != NULL; ptr = ptr->next)
            ;
        ptr->next = h2;
    }
}
void simplify(struct node **h)
{
    struct node *ptr, *ptr1, *prev;
    ptr = *h;
    while (ptr != NULL)
    {
        prev = ptr;
        ptr1 = ptr->next;
        while (ptr1 != NULL)
        {
            if (ptr1->exp == ptr->exp)
            {
                ptr->cof += ptr1->cof;
                prev->next = ptr1->next;
                free(ptr1);
                ptr1 = prev;
            }
            prev = ptr1;
            ptr1 = ptr1->next;
        }
        ptr = ptr->next;
    }
}
void display(struct node *h)
{
    struct node *ptr;
    ptr = h;
    if (h == NULL)
    {
        printf("list is empty");
        return;
    }
    for (; ptr != NULL; ptr = ptr->next)
    {
        if (ptr->cof >= 0)
        {
            if (ptr != NULL)
                printf("+");
        }
        printf("%dX^%d\t", ptr->cof, ptr->exp);
    }
    printf("\n\n");
}
```

```
int main()
{
   struct node *h1, *h2;
   h1 = h2 = NULL;
   create(&h1);
   create(&h2);
   printf("the 1st polynoial is\n");
   display(h1);
   printf("the 2nd polynoial is\n");
   display(h2);
   join(&h1, h2);
   simplify(&h1);
   printf("the polynomial after adding is\n");
   display(h1);
}
```

```
enter the no. of terms in the polynomial 3
enter the coefficient and the exponent of the 1th term 2 5
enter the coefficient and the exponent of the 2th term 9 7
enter the coefficient and the exponent of the 3th term 3 3
enter the no. of terms in the polynomial 2
enter the coefficient and the exponent of the 1th term 25 7
enter the coefficient and the exponent of the 2th term 9 10
the 1st polynoial is
+2X^5   +9X^7   +3X^3

the 2nd polynoial is
+25X^7  +9X^10

the polynomial after adding is
+2X^5   +34X^7  +3X^3   +9X^10

PS D:\KIIT_NOTES\2nd year sem_3\dsa_lab\7_9_2021>
```

## *Question3 : A matrix m × n that has relatively few non-zero entries is called sparse matrix. It may be represented in much less than m × n space. An m × n matrix with k non-zero entries is sparse if k << m × n. It may be faster to represent the matrix compactly as a list of the non-zero indexes and associated entries. WAP to represent a sparse matrix in 3-tuple format by using array.*

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

struct Node
{
   int row;
   int col;
   int data;
   struct Node *next;
};
```

```
struct Node *insertNodeAtTail(struct Node *head, int r, int c, int d)
{
   struct Node *ptr = (struct Node *)malloc(sizeof(struct Node));
   ptr->row = r;
   ptr->col = c;
   ptr->data = d;
   ptr->next = NULL;
   if (head == NULL)
   {
      head = ptr;
   }
   else
   {
      struct Node *p = head;
      while (p->next != NULL)
      {
         p = p->next;
      }
      p->next = ptr;
   }
   return head;
}

void display(struct Node *head)
{
   struct Node *ptr = head;

   printf("Printing the matrix in triplet form :\n");
   printf("Row   Column  Element \n");
   while (ptr != NULL)
   {
      printf("%d\t%d\t%d\n", ptr->row, ptr->col, ptr->data);
      ptr = ptr->next;
   }
}

int main()
{
   struct Node *head = NULL;
   int row, col, n, i, r, c, num;
   printf("Enter the dimension of the matrix :\n");
   printf("Enter the no. of row :\n");
   scanf("%d", &row);
   printf("Enter the no. of columns :\n");
   scanf("%d", &col);
   printf("Enter the number of non-zero inputs to be given to the matrix :\n");
   scanf("%d", &n);
   if (n < (row * col) / 2)
   {
      for (i = 0; i < n; i++)
      {
         printf("Enter the row index :\n");
         scanf("%d", &r);

         printf("Enter the column index :\n");
         scanf("%d", &c);

         printf("Enter the element :\n");
```

```
            scanf("%d", &num);

            head = insertNodeAtTail(head, r, c, num);
        }
        display(head);
    }
    else
    {
        printf("Matrix is not sparse\n");
    }
    return 0;
}
```

```
Enter the dimension of the matrix :
Enter the no. of row :
2
Enter the no. of columns :
3
Enter the number of non-zero inputs to be given to the matrix :
2
Enter the row index :
0
Enter the column index :
0
Enter the element :
27
Enter the row index :
1
Enter the column index :
2
Enter the element :
99
Printing the matrix in triplet form :
Row   Column  Element
0       0       27
1       2       99
PS D:\KIIT_NOTES\2nd year sem_3\dsa_lab\7_9_2021>
```