**Assignment No. 8**

**AIM**:- Impliment polynomial using circular linked list and perform

i: Addition/multiplication of polynomials

Ii:Evaluation of polynomials

**OBJECTIVE**:-

Circular lists are the required data structure when we want a list to be accessed in a circle or loop. Address reference to previous node can easily be found which is not posiible in single linked list.from this polynomial arithmatic using circular linked lists we can understand accessing and handling the data of circular linked lists and perform different operations.

**Theory** :-

1. Circular linked list is a linked list where all nodes are connected to form a circle. There is no NULL at the end. A circular linked list can be a singly circular linked list or doubly circular linked list. ... We can maintain a pointer to the last inserted node and front can always be obtained as next of last.

2. Following are the important operations supported by a circular list.

insert − Inserts an element at the start of the list.

delete − Deletes an element from the start of the list.

display − Displays the list.

3. A polynomial p(x) is the expression in variable x which is in the form (axn + bxn-1 + …. + jx+ k), where a, b, c …., k fall in the category of real numbers and 'n' is non negative integer, which is called the degree of polynomial. An essential characteristic of the polynomial is that each term in the polynomial expression consists of two parts:

* One is the coefficient
* other is the exponent

4. A polynomial can be thought of as an ordered list of non zero terms. Each non zero term is a two-tuple which holds two pieces of information:

* The exponent part
* The coefficient part

**Algorithm :-**

1. **Create circular linked list by including coef,exp,and pointer to next varialble in structure definition.**
2. **For each node ask user to input coef and exp and save them in particular node.**
3. **For displaying polynomial traverse through each node of circular linked list and display in form of (coef)x^(exp)**
4. **For addition traverse through the cll and check the exp first if the exp match then add the coef and**

**Display the resulting polynomial else skip to next node.**

1. **For subtraction traverse through the cll and check the exp first if the exp match then subtract the coef and**

**Display the resulting polynomial skip to next node.**

1. **For multiplication for each node multiply their coef and add their exp.**

**Sourcecode :-**

**#include<iostream>**

**#include<math.h>**

**#include<stdlib.h>**

**using namespace std;**

**struct node{**

**int exp;**

**int p;**

**struct node \*nxt;**

**};**

**typedef struct node \*NODE;**

**NODE attach(int ex,int po,NODE hd){ //ATTACH POLY**

**NODE nn,temp;**

**nn= new node;**

**nn->exp = ex;**

**nn->p = po;**

**temp = hd;**

**while(temp->nxt != hd)**

**{**

**temp = temp->nxt;**

**}**

**temp->nxt = nn;**

**nn->nxt=hd;**

**return hd;**

**}**

**NODE read(NODE hd){ //INPUT POLY**

**int i=1,coef,pow;**

**cout<<"Enter -999 as coefficient to end polynomial.\n";**

**while(1){**

**cout<<"Enter the "<<i<<" coefficient: ";**

**cin>>coef;**

**if(coef == -999)**

**break;**

**cout<<"Enter its power: ";**

**cin>>pow;**

**hd = attach(coef,pow,hd);**

**i++;**

**}**

**return hd;**

**}**

**NODE poly\_add(NODE head1, NODE head2, NODE head3)**

**{**

**NODE a,b;**

**int coeff;**

**a = head1->nxt;**

**b = head2->nxt;**

**while(a != head1 && b != head2)**

**{**

**if(a->p == b->p)**

**{**

**coeff = a->exp + b->exp;**

**if(coeff != 0)**

**head3 = attach(coeff, a->p, head3);**

**a = a->nxt;**

**b = b->nxt;**

**}**

**else if(a->p > b->p)**

**{**

**head3 = attach(a->exp, a->p, head3);**

**a = a->nxt;**

**}**

**else**

**{**

**head3 = attach(b->exp, b->p, head3);**

**b = b->nxt;**

**}**

**}**

**while(a != head1)**

**{**

**head3 = attach(a->exp, a->p, head3);**

**a = a->nxt;**

**}**

**while(b != head2)**

**{**

**head3 = attach(b->exp, b->p, head3);**

**b = b->nxt;**

**}**

**NODE c;**

**NODE d;**

**NODE f;**

**c = head3->nxt;**

**while(c != head3)**

**{**

**NODE e;**

**d = c;**

**e = c->nxt;**

**while(e != head3)**

**{**

**if(c->p == e->p)**

**{**

**c->exp = c->exp+e->exp;**

**d->nxt = e->nxt;**

**delete(e);**

**break;**

**}**

**else{**

**d = e;**

**e = e->nxt;**

**}**

**}**

**c = c->nxt;**

**}**

**return head3;**

**}**

**void disp(NODE hd)**

**{**

**NODE tmp;**

**if(hd->nxt == hd){**

**cout<<"\nThe Polynomial doesn't exist.\n";**

**return;**

**}**

**tmp= hd->nxt;**

**while(tmp != hd)**

**{**

**cout<<tmp->exp<<"x^"<<tmp->p;**

**tmp= tmp->nxt;**

**if(tmp != hd && (tmp->exp)>0)**

**cout<<"+";**

**}**

**}**

**void eval(NODE hd){**

**int sum=0,x=0,co=0;**

**NODE tm;**

**if(hd->nxt == hd){**

**cout<<"\nEmpty polynomial.\n";**

**return;}**

**tm = hd->nxt;**

**cout<<"\nEnter Value of x\n";**

**cin>>x;**

**while(tm != hd){**

**co = pow(x,tm->p);**

**sum = sum + (tm->exp \* co);**

**tm = tm->nxt;**

**}**

**cout<<"Evaluted polymonial answer: "<<sum<<"\n";**

**}**

**int main(){**

**NODE p1,p2,p3;**

**int ch=0;**

**p1= new node;**

**p2=new node;**

**p3=new node;**

**p1->nxt=p1;**

**p2->nxt=p2;**

**p3->nxt=p3;**

**cout<<"Enter first polynomial.\n";**

**p1 = read(p1);**

**cout<<"Enter second polynomial.\n";**

**p2 = read(p2);**

**do{**

**cout<<"\n\nEnter 1 for displaying entered polynomials.\n";**

**cout<<"Enter 2 to add the polynomials and display result.\n";**

**cout<<"Enter 3 to evaluate the answer.\n";**

**cout<<"Enter 4 to exit.\n";**

**cin>>ch;**

**switch(ch){**

**case 1:**

**system("clear");**

**cout<<"\nFirst polynomial:\n";**

**disp(p1);**

**cout<<"\nSecond polynomial:\n";**

**disp(p2);**

**break;**

**case 2:**

**system("clear");**

**p3 = poly\_add(p1,p2,p3);**

**cout<<"\nAdded polynomial:\n";**

**disp(p3);**

**cout<<"\n";**

**break;**

**case 3:**

**system("clear");**

**disp(p3);**

**eval(p3);**

**cout<<"\n";**

**break;**

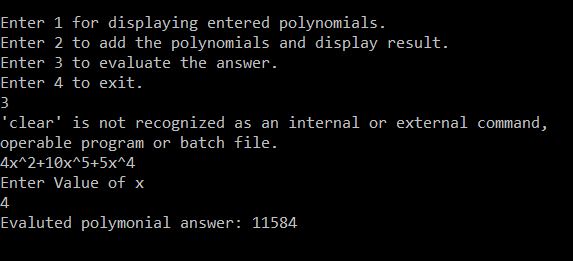
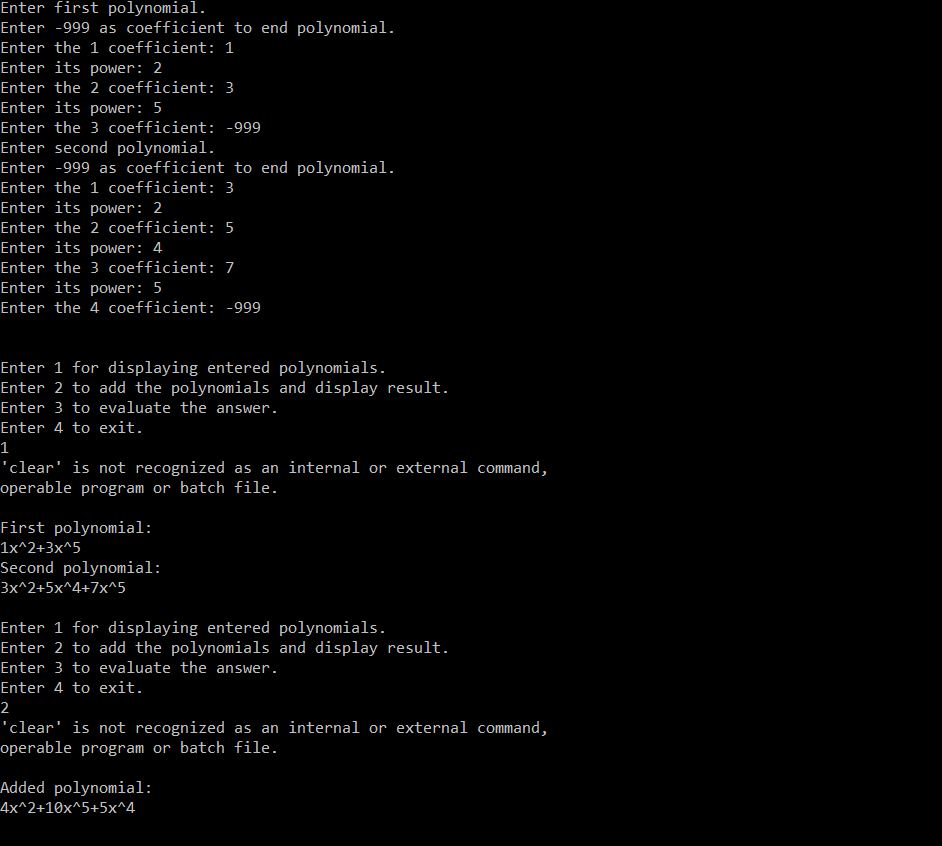
**}**

**}**

**while(ch != 4);**

**return 0;**

**}**

****