Bansilal Ramnath Agarwal Charitable Trust's

Vishwakarma Institute of Information Technology

Department of
Artificial Intelligence and Data Science

Name: Siddhesh Dilip Khairnar

Class: SY

Division: B

Roll No: 272028

Semester: III

Academic Year: 2022-2023

Subject Name & Code: Data Structure, ADUA21202

Title of Assignment: Implement polynomial using Doubly Linked List and perform Addition of Polynomials

Assignment No.- 4

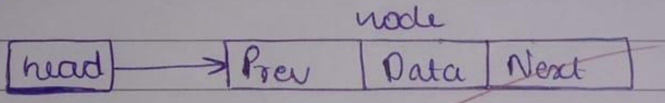
DS Assignment : 4

Name : siddhesh Dilip khairnar
PRN NO : 22110398 ~~PRN no:~~
Roll no : 272028

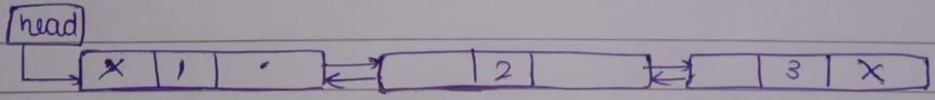
Problem statment: Implement polynomial using Doubly linked list & peform Addition / Multiplication of polynomial.

Theory:

Doubly linked list: Doubly linked list is a complex type of linked list in which a node contais a pointer to the previous as well as the next. node in the sequence. Therefore, in a doubly linked list, a node consist of three part : node data, pointer to the next node in sequence, pointer to the previous node. A sample node in a doubly linked list is shown in the figure :



A doubly linked list containing three node having number from 1 to 3 in their data, is shown in the following image



In C, structure of a node in doubly linked list can be given as :

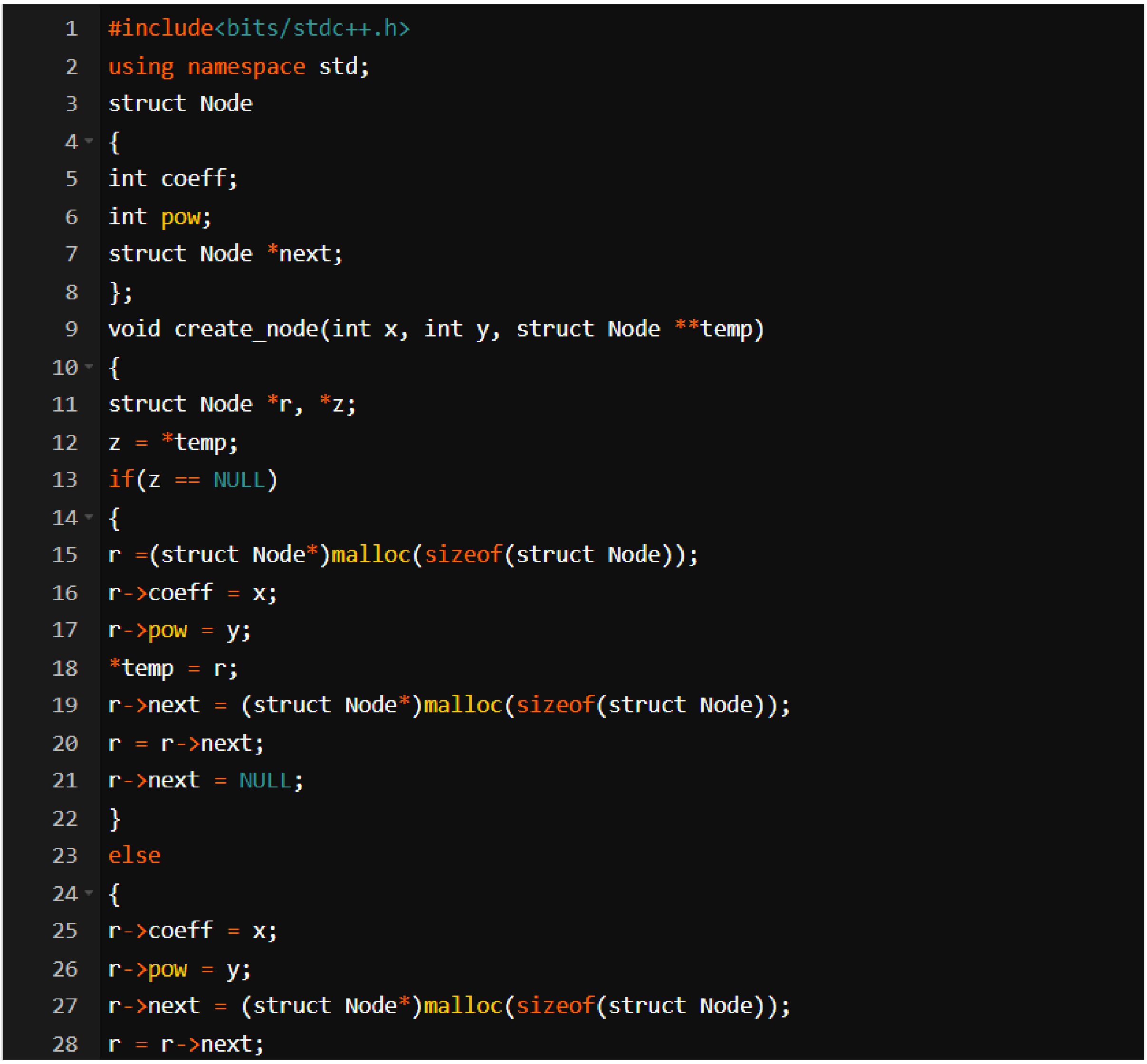
```
struct node
{   struct node * Prev;
    int data;
    struct node * next;
}
```

Conclusion : Thus, we have succesfully implemented polynomial using Doubly linked list and performed Addition / Multiplication of polynomials.

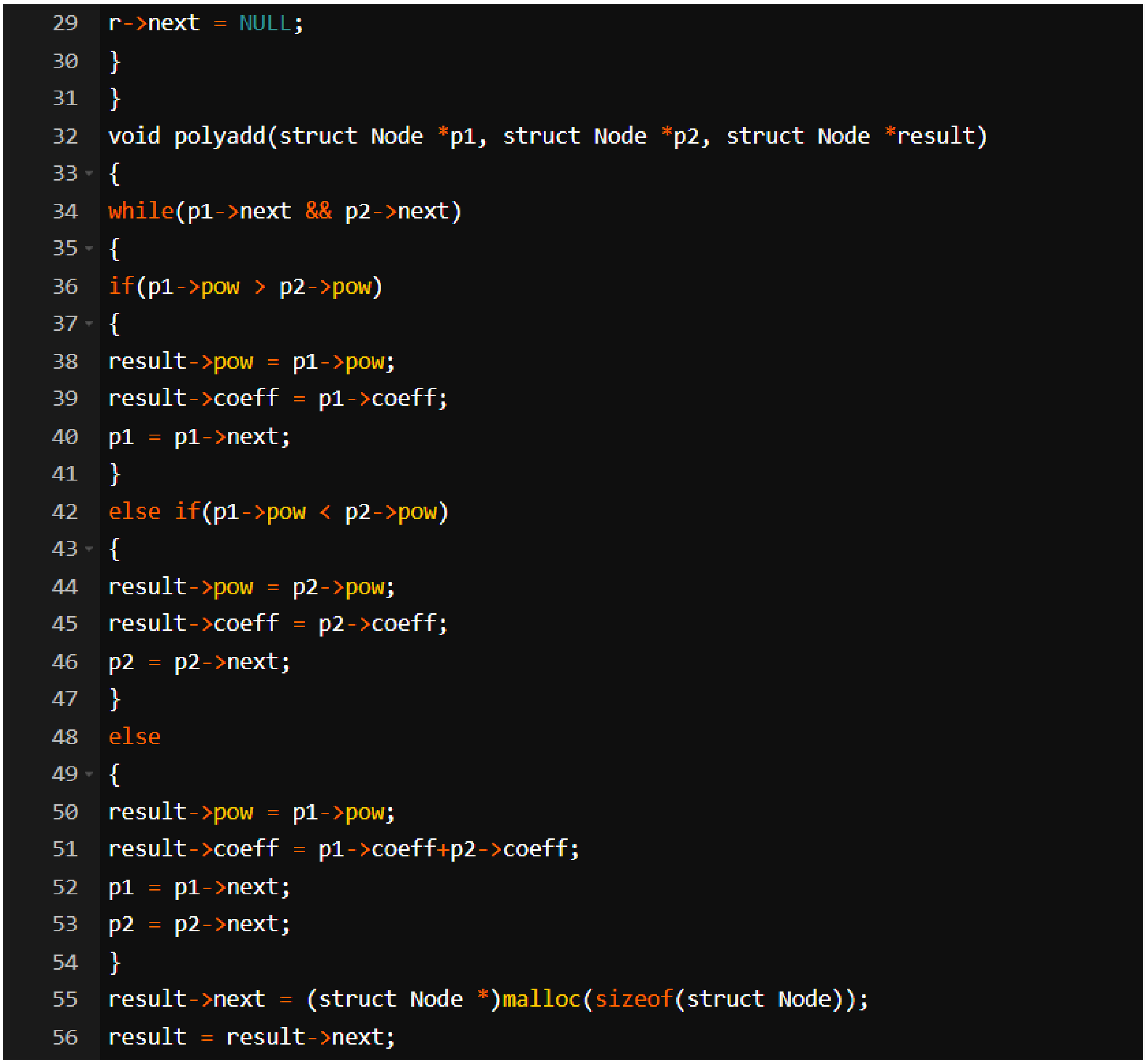
c

Aswin

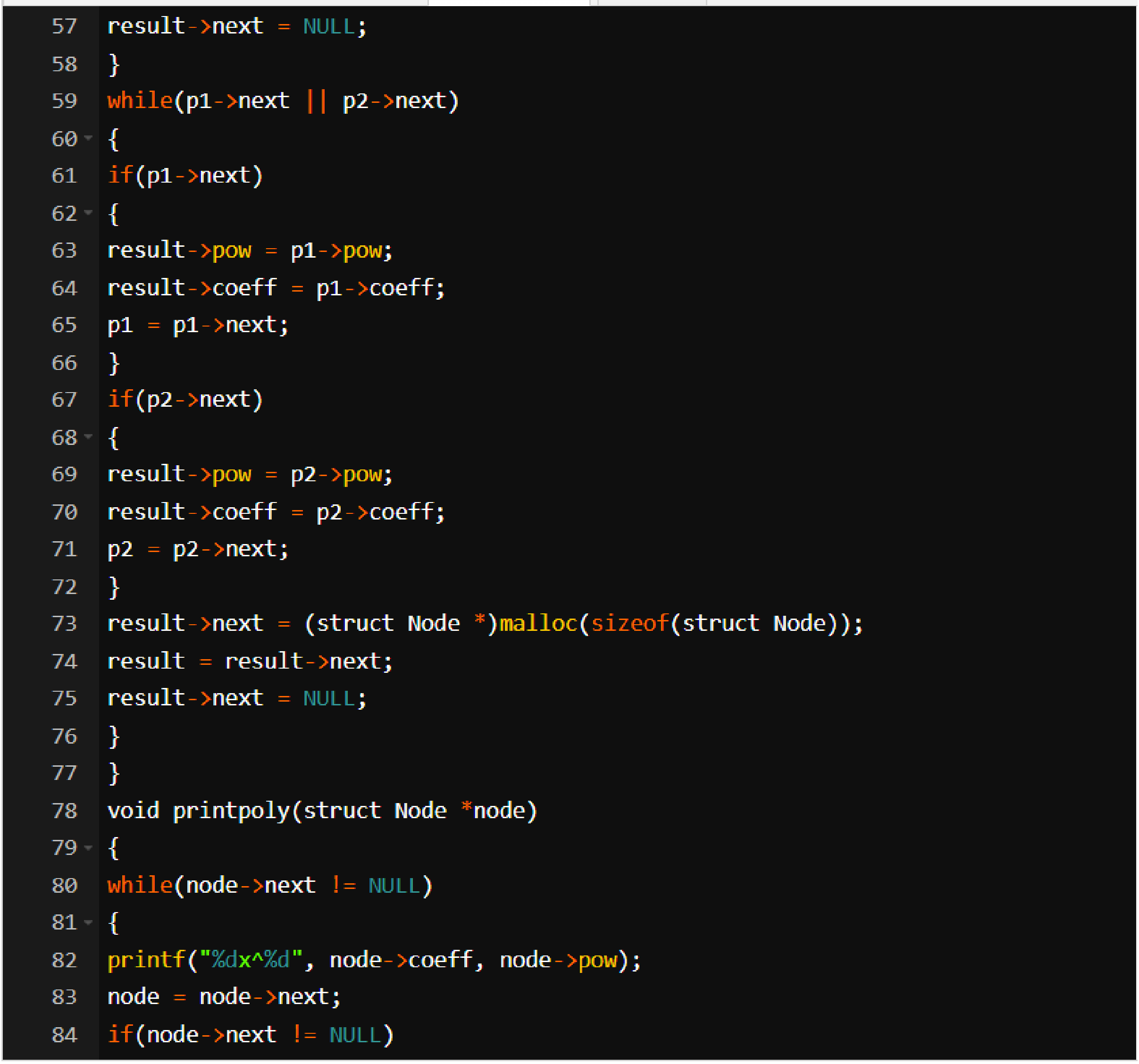
Program:



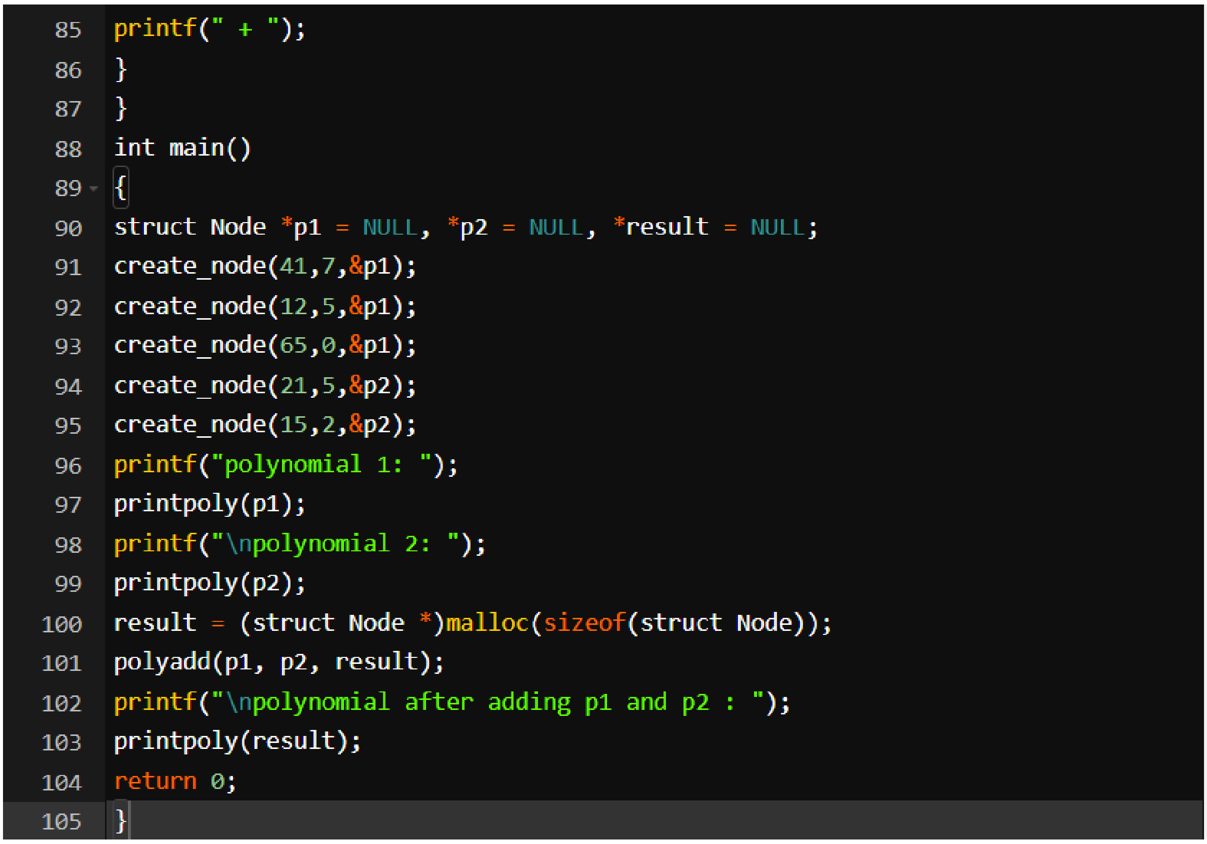
```
#include<bits/stdc++.h>
using namespace std;
struct Node
{
int coeff;
int pow;
struct Node *next;
};
void create_node(int x, int y, struct Node **temp)
{
struct Node *r, *z;
z = *temp;
if(z == NULL)
{
r =(struct Node*)malloc(sizeof(struct Node));
r->coeff = x;
r->pow = y;
*temp = r;
r->next = (struct Node*)malloc(sizeof(struct Node));
r = r->next;
r->next = NULL;
}
else
{
r->coeff = x;
r->pow = y;
r->next = (struct Node*)malloc(sizeof(struct Node));
r = r->next;
```



```
r->next = NULL;
}
}
void polyadd(struct Node *p1, struct Node *p2, struct Node *result)
{
while(p1->next && p2->next)
{
if(p1->pow > p2->pow)
{
result->pow = p1->pow;
result->coeff = p1->coeff;
p1 = p1->next;
}
else if(p1->pow < p2->pow)
{
result->pow = p2->pow;
result->coeff = p2->coeff;
p2 = p2->next;
}
else
{
result->pow = p1->pow;
result->coeff = p1->coeff+p2->coeff;
p1 = p1->next;
p2 = p2->next;
}
result->next = (struct Node *)malloc(sizeof(struct Node));
result = result->next;
```

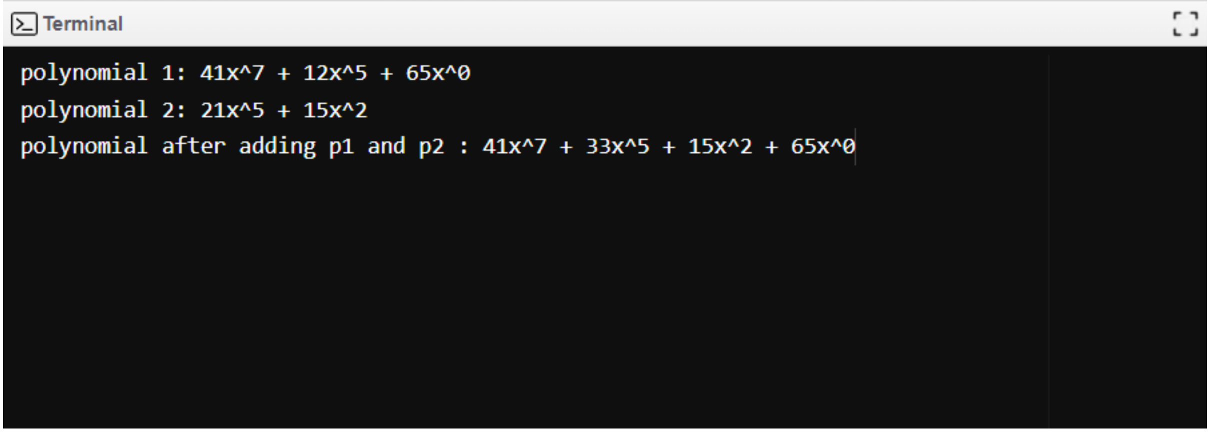


```
result->next = NULL;
}
while(p1->next || p2->next)
{
if(p1->next)
{
result->pow = p1->pow;
result->coeff = p1->coeff;
p1 = p1->next;
}
if(p2->next)
{
result->pow = p2->pow;
result->coeff = p2->coeff;
p2 = p2->next;
}
result->next = (struct Node *)malloc(sizeof(struct Node));
result = result->next;
result->next = NULL;
}
}
void printpoly(struct Node *node)
{
while(node->next != NULL)
{
printf("%dx^%d", node->coeff, node->pow);
node = node->next;
if(node->next != NULL)
```



```
printf(" + ");
}
}
int main()
{
struct Node *p1 = NULL, *p2 = NULL, *result = NULL;
create_node(41,7,&p1);
create_node(12,5,&p1);
create_node(65,0,&p1);
create_node(21,5,&p2);
create_node(15,2,&p2);
printf("polynomial 1: ");
printpoly(p1);
printf("\npolynomial 2: ");
printpoly(p2);
result = (struct Node *)malloc(sizeof(struct Node));
polyadd(p1, p2, result);
printf("\npolynomial after adding p1 and p2 : ");
printpoly(result);
return 0;
}
```

Output:



```
polynomial 1: 41x^7 + 12x^5 + 65x^0
polynomial 2: 21x^5 + 15x^2
polynomial after adding p1 and p2 : 41x^7 + 33x^5 + 15x^2 + 65x^0
```