```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
struct node
{
    float cf; float px, py;
    int flag;
    struct node *link;
};
typedef struct node *NODE;
  NODE getnode()
   {
     NODE x;
    x=(NODE) malloc (sizeof (struct node));
    if (x==NULL)
    {
      pf("mem full\n");
       exit (0);
     }
    return x;
}
NODE insert_rear (float cf, float x, float y, NODE fi
   { NODE temp, cur;
     temp= getnode();
     temp->cf = cf;
      temp -> px= x;
     temp -> link= NULL;
      temp-> flag= 0;
     if (first == NULL).
```

```
    } return temp; }
cur = first;
while (cur->link != NULL)
    {
        cur = cur->link;
    }
cur->link = temp;
return first;
}
NODE read_poly(NODE first)
{
    int i;
    float cf, px, py;
printf("Enter -999 to end the polynomial:\n");
for (i=1;;i++)
    {
    printf("Enter %d term:", i);
    printf("Co-efficient");
scanf("%f", &cf);
    if(cf==-999)
    { break; }
    printf("Power of x:");
    scanf("%d", &px);
    printf("Power of y:");
    scanf("%d", &py);
first = insert_rear(cf, px, py, first);
    }
    return first;
}
```

```
float evaluate_polynomial(NODE first)
{
    float x, y, sum=0;
    NODE polynomial;
    pf("Enter the values of x & y:\n");
    sf("%f%f", &x, &y);
    polynomial = first;
    while (polynomial != NULL)
{
sum = sum + polynomial->cf * pow(x, polynomial->px)
        * pow(y, polynomial->py);
        polynomial = polynomial->link;
    }
    return sum;
}

void display(NODE first)
{ NODE temp;
    if (first == NULL)
    { pf("Polynomial does not exist");
    }
else
{ temp = first;
    while (temp->link != NULL)
{ pf("(%.5.2f x^%.3.2f y^%.3.2f)\t+", temp->cf,
            temp->px, temp->py);
        temp = temp->link;
    }
pf("(%.5.2f x^%.3.2f y^%.3.2f)\n", temp->cf, temp->
                                    temp->py);
    }
}
```

```
int main()
{ NODE first;
  float res;
  first = NULL;
  printf("Enter the polynomial: \n");
  first = read_poly(first);
  res = evaluate_polynomial(first);
  printf("Polynomial is: \n");
  display(first);
  printf("Result is = %f \n", res);
}
```