```
                                Read co
                                Display “Enter exponent for polynomial 2”
                                Read exp
                                Call create(p2, co, exp)
                        Case 3:
                                Set sum <- call polyAdd(p1, p2, sum)
                                Call display(sum)
                        Case 4:
                                Set flag <- 0
                End switch

Step 7: Stop
```

## Code

C | C++ | Java | Python

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

//polynomial node structure
struct node
{
    int co, exp;
    struct node* next;
};

//create a polynomial
struct node* create(struct node* head, int co, int exp)
{
    struct node *temp, *flag;
//if polynomial empty. make the node the head node

    if(head == NULL)
    {
     temp = (struct node*) malloc (sizeof(struct node));
     temp->co = co;
     temp->exp = exp;
     temp->next = NULL;
     head = temp;
    }
    else
    {
     //else go to the last node and append
     temp = head;
     while(temp->next != NULL)
         temp = temp->next;
     flag = (struct node *)malloc(sizeof(struct node));
     flag->co = co;
     flag->exp = exp;
```

```
     flag->next = NULL;
     temp->next = flag;
    }

    return head;
}

//add two polynomial
struct node* polyAdd(struct node *p1, struct node *p2, struct node *sum)
{
    //copy the two polynomial and initialize variable res to store the sum
    struct node *poly1 = p1, *poly2 = p2, *res;

    //if polynomial 2 is null, set polynomial 1 as the sum
    if(poly1 != NULL && poly2 == NULL)
    {
     sum = poly1;
     return sum;
    }

    //if polynomial 1 is null, set polynomial 2 as the sum
    else if(poly1 == NULL && poly2 != NULL)
    {
     sum = poly2;
     return sum;
    }

    //if both polynomials are non-empty
    while(poly1 != NULL && poly2 != NULL)
    {
     //if the sum is empty, initialize sum with a node structure
     //and set res equal to sum
     if(sum == NULL)
     {
         sum = (struct node *)malloc(sizeof(struct node));
         res = sum;
     }

     //add a new node structure at the end of res to store sum
     else
     {
         res->next = (struct node *)malloc(sizeof(struct node));;
         res = res->next;
     }

     //if exponent of current node of polynomial 1 is greater than that of polynomial 2
     //add it to the sum
     if(poly1->exp > poly2->exp)
     {
```

```
        res->co = poly1->co;
        res->exp = poly1->exp;
        poly1 = poly1->next;
     }

     //if exponent of current node of polynomial 2 is greater than that of polynomial 1
     //add it to the sum
     else if(poly1->exp < poly2->exp)
     {
        res->co = poly2->co;
        res->exp = poly2->exp;
        poly2 = poly2->next;
     }

     //if exponent of current node of polynomial 1 is equal to that of polynomial 2
     //add the sum of their co-efficient to the sum
     else if(poly1->exp == poly2->exp)
     {
        res->co = poly1->co + poly2->co;
        res->exp = poly1->exp;
        poly1 = poly1->next;
        poly2 = poly2->next;
     }
    }

    //if polynomial 1 is non-empty add the remaining nodes to the sum
    while(poly1 != NULL)
    {

     res->next = (struct node *)malloc(sizeof(struct node));;
     res = res->next;

     res->co = poly1->co;
     res->exp = poly1->exp;
     poly1 = poly1->next;
    }

    //if polynomial 2 is non-empty add the remaining nodes to the sum
    while(poly2 != NULL)
    {
     res->next = (struct node *)malloc(sizeof(struct node));;
     res = res->next;

     res->co = poly2->co;
     res->exp = poly2->exp;
     poly2 = poly2->next;
    }

    //set pointer of last node to null
```

```
    res->next = NULL;

    //return the head node of the sum
    return sum;
}

//display polynomial
void display(struct node* head)
{
    struct node *temp=head;
    while(temp != NULL)
    {
     printf("%d^%d+", temp->co, temp->exp);
     temp=temp->next;
    }
    printf("\n");
}

void main()
{
    //to store polynomial 1, polynomial 2 and the sum
    struct node *p1 = NULL, *p2 = NULL, *sum = NULL;
    int ch, co, exp;
    int loop = 1;
    while(loop) {
     printf("1. Add to Polynomial 1\n");
     printf("2. Add to Polynomial 1\n");
     printf("3. Perform Addition\n");
     printf("4. Exit\n");
     scanf("%d", &ch);
     switch(ch)
     {
         case 1: printf("Enter co-efficient\n");
                 scanf("%d", &co);
                 printf("Enter exponent\n");
                 scanf("%d", &exp);
                 p1 = create(p1, co, exp);
                 break;

         case 2: printf("Enter co-efficient\n");
                 scanf("%d", &co);
                 printf("Enter exponent\n");
                 scanf("%d", &exp);
                 p2 = create(p2, co, exp);
                 break;
         case 3: sum=polyAdd(p1,p2,sum);
                 printf("\nPolynomial 1\n");
                 display(p1);
                 printf("\nPolynomial 2\n");
```

```
                display(p2);
                printf("\nSum:\n");
                display(sum);
                break;
        case 4: loop = 0;
                break;
        default: printf("Wrong Choice! Re-enter\n");
                break;
     }
    }
}
```

## Output

```
1. Enter Polynomial 1
2. Enter Polynomial 2
3. Perform Addition
4. Exit
1
Enter co-efficient
5
Enter exponent
4

1. Enter Polynomial 1
2. Enter Polynomial 2
3. Perform Addition
4. Exit
1
Enter co-efficient
2
Enter exponent
2

1. Enter Polynomial 1
2. Enter Polynomial 2
3. Perform Addition
4. Exit
1
Enter co-efficient
3
Enter exponent
1

1. Enter Polynomial 1
2. Enter Polynomial 2
3. Perform Addition
4. Exit
```