

```

        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](#)

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 //polynomial node structure
5 struct node
6 {
7     int co, exp;
8     struct node* next;
9 };
10
11 //create a polynomial
12 struct node* create(struct node* head, int co, int exp)
13 {
14     struct node *temp, *flag;
15     //if polynomial empty. make the node the head node
16
17     if(head == NULL)
18     {
19         temp = (struct node*) malloc (sizeof(struct node));
20         temp->co = co;
21         temp->exp = exp;
22         temp->next = NULL;
23         head = temp;
24     }
25     else
26     {
27         //else go to the last node and append
28         temp = head;
29         while(temp->next != NULL)
30             temp = temp->next;
31         flag = (struct node *)malloc(sizeof(struct node));
32         flag->co = co;
33         flag->exp = exp;

```

```

34     flag->next = NULL;
35     temp->next = flag;
36 }
37
38     return head;
39 }
40
41 //add two polynomial
42 struct node* polyAdd(struct node *p1, struct node *p2, struct node *sum)
43 {
44     //copy the two polynomial and initialize variable res to store the sum
45     struct node *poly1 = p1, *poly2 = p2, *res;
46
47     //if polynomial 2 is null, set polynomial 1 as the sum
48     if(poly1 != NULL && poly2 == NULL)
49     {
50         sum = poly1;
51         return sum;
52     }
53
54     //if polynomial 1 is null, set polynomial 2 as the sum
55     else if(poly1 == NULL && poly2 != NULL)
56     {
57         sum = poly2;
58         return sum;
59     }
60
61     //if both polynomials are non-empty
62     while(poly1 != NULL && poly2 != NULL)
63     {
64         //if the sum is empty, initialize sum with a node structure
65         //and set res equal to sum
66         if(sum == NULL)
67         {
68             sum = (struct node *)malloc(sizeof(struct node));
69             res = sum;
70         }
71
72         //add a new node structure at the end of res to store sum
73         else
74         {
75             res->next = (struct node *)malloc(sizeof(struct node));
76             res = res->next;
77         }
78
79         //if exponent of current node of polynomial 1 is greater than that of polynomial 2
80         //add it to the sum
81         if(poly1->exp > poly2->exp)
82         {

```

```

83         res->co = poly1->co;
84         res->exp = poly1->exp;
85         poly1 = poly1->next;
86     }
87
88     //if exponent of current node of polynomial 2 is greater than that of polynomial 1
89     //add it to the sum
90     else if(poly1->exp < poly2->exp)
91     {
92         res->co = poly2->co;
93         res->exp = poly2->exp;
94         poly2 = poly2->next;
95     }
96
97     //if exponent of current node of polynomial 1 is equal to that of polynomial 2
98     //add the sum of their co-efficient to the sum
99     else if(poly1->exp == poly2->exp)
100    {
101        res->co = poly1->co + poly2->co;
102        res->exp = poly1->exp;
103        poly1 = poly1->next;
104        poly2 = poly2->next;
105    }
106 }
107
108 //if polynomial 1 is non-empty add the remaining nodes to the sum
109 while(poly1 != NULL)
110 {
111
112     res->next = (struct node *)malloc(sizeof(struct node));;
113     res = res->next;
114
115     res->co = poly1->co;
116     res->exp = poly1->exp;
117     poly1 = poly1->next;
118 }
119
120 //if polynomial 2 is non-empty add the remaining nodes to the sum
121 while(poly2 != NULL)
122 {
123     res->next = (struct node *)malloc(sizeof(struct node));;
124     res = res->next;
125
126     res->co = poly2->co;
127     res->exp = poly2->exp;
128     poly2 = poly2->next;
129 }
130
131 //set pointer of last node to null

```

```
132     res->next = NULL;
133
134     //return the head node of the sum
135     return sum;
136 }
137
138 //display polynomial
139 void display(struct node* head)
140 {
141     struct node *temp=head;
142     while(temp != NULL)
143     {
144         printf("%d^%d+", temp->co, temp->exp);
145         temp=temp->next;
146     }
147     printf("\n");
148 }
149
150 void main()
151 {
152     //to store polynomial 1, polynomial 2 and the sum
153     struct node *p1 = NULL, *p2 = NULL, *sum = NULL;
154     int ch, co, exp;
155     int loop = 1;
156     while(loop) {
157         printf("1. Add to Polynomial 1\n");
158         printf("2. Add to Polynomial 1\n");
159         printf("3. Perform Addition\n");
160         printf("4. Exit\n");
161         scanf("%d", &ch);
162         switch(ch)
163         {
164             case 1: printf("Enter co-efficient\n");
165                     scanf("%d", &co);
166                     printf("Enter exponent\n");
167                     scanf("%d", &exp);
168                     p1 = create(p1, co, exp);
169                     break;
170
171             case 2: printf("Enter co-efficient\n");
172                     scanf("%d", &co);
173                     printf("Enter exponent\n");
174                     scanf("%d", &exp);
175                     p2 = create(p2, co, exp);
176                     break;
177             case 3: sum=polyAdd(p1,p2,sum);
178                     printf("\nPolynomial 1\n");
179                     display(p1);
180                     printf("\nPolynomial 2\n");
```

```

181         display(p2);
182         printf("\nSum:\n");
183         display(sum);
184         break;
185     case 4: loop = 0;
186         break;
187     default: printf("Wrong Choice! Re-enter\n");
188         break;
189 }
190 }
191 }

```

## 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

```