

MIT WORLD PEACE UNIVERSITY

Advanced Data Structures  
Second Year B. Tech, Semester 4

---

---

ADDITION OF POLYNOMIALS USING CIRCULAR  
LINKED LISTS

---

---

ASSIGNMENT No. 1

Prepared By

Krishnaraj Thadesar  
Cyber Security and Forensics  
Batch A1, PA 20

January 18, 2023

# Contents

<b>1 Aim</b>	<b>1</b>
<b>2 Objectives</b>	<b>1</b>
<b>3 Problem Statement</b>	<b>1</b>
<b>4 Theory</b>	<b>1</b>
<b>5 Platform</b>	<b>1</b>
<b>6 Pseudo Code or Algorithm</b>	<b>1</b>
<b>7 Input</b>	<b>1</b>
<b>8 Output</b>	<b>1</b>
<b>9 Code</b>	<b>1</b>
9.1 Program . . . . .	1
9.2 Input and Output . . . . .	5
<b>10 Conclusion</b>	<b>7</b>
<b>11 FAQ</b>	<b>8</b>

## 1 Aim

## 2 Objectives

## 3 Problem Statement

## 4 Theory

## 5 Platform

**Operating System:** Arch Linux x86-64

**IDEs or Text Editors Used:** Visual Studio Code

**Compilers :** g++ and gcc on linux for C++

## 6 Pseudo Code or Algorithm

## 7 Input

1. The Choice for what to do
2. The Coefficients and Exponents of the Polynomials

## 8 Output

1. The Resultant Polynomial Represented as a Circular Linked List
2. The Sum of the Given 2 Polynomials.
3. The Menu for what to do.

## 9 Code

### 9.1 Program

```
1 // Circular linked List
2 // Implementing the following polynomial operations using circular linked list.
  Create Dislay and Add.
3
4 // Krishnaraj Thadesar
5 // Jan 22nd 2023
6 // Assignment 1
7 // Advanced Data Structures
8 // Semester 4
9
10 #include <stdio.h>
11 #include <stdlib.h>
12
13 // The default way to represent circular linked lists using structures.
14 struct Node
```

```
15 {
16     int coeff;
17     int exp;
18     struct Node *next;
19 };
20
21 // Function to accept the polynomial from the user.
22 void accept_polynomial(struct Node *head)
23 {
24     struct Node *temp = head;
25     int choice = 0;
26     do
27     {
28         struct Node *curr = (struct Node *)malloc(sizeof(struct Node));
29         printf("\nEnter the Coefficient: ");
30         scanf("%d", &curr->coeff);
31         printf("\nEnter the Exponent: ");
32         scanf("%d", &curr->exp);
33
34         // Main Logic of inserting node at the end and making it point to the head
35         curr->next = head;
36         temp->next = curr;
37         temp = temp->next;
38
39         printf("Do you want to enter more terms? (0 for no, 1 for yes) \n");
40         scanf("%d", &choice);
41     } while (choice != 0);
42 }
43
44 // Function to display the polynomial.
45 int display_polynomial(struct Node *head)
46 {
47     // Edge Case of empty list.
48     if (head->next == head)
49     {
50         printf("\nNo terms in the polynomial");
51         return -1;
52     }
53
54     struct Node *curr = (struct Node *)malloc(sizeof(struct Node));
55     curr = head->next;
56
57     while (curr != head)
58     {
59         printf("%dx^%d", curr->coeff, curr->exp);
60         curr = curr->next;
61         if (curr != head)
62         {
63             printf(" + ");
64         }
65     }
66     printf("\n");
67 }
68
69 // Function to add two polynomials, Returns the head of the added polynomial.
70 // Takes as input the heads of the other 2 polynomials that you want to add.
71 struct Node *add_polynomials(struct Node *head1, struct Node *head2)
72 {
```

```
73 // Pointers for the result polynomial.
74 struct Node *result_head = (struct Node *)malloc(sizeof(struct Node));
75 result_head->next = result_head;
76 struct Node *result_temp = result_head;
77 struct Node *result_current;
78
79 // p1 and p2 are the pointers to the first node of the two polynomials.
80 struct Node *p1 = head1->next;
81 struct Node *p2 = head2->next;
82
83 // In case one of the polynomial exhausts before the other one.
84 while (p1 != head1 && p2 != head2)
85 {
86     // if the exponents are equal, add the coefficients and add the node to
the result polynomial.
87     if (p1->exp == p2->exp)
88     {
89         // Copy the data of the sum of the nodes to the result polynomial.
90         result_current = (struct Node *)malloc(sizeof(struct Node));
91         result_current->coeff = p1->coeff + p2->coeff;
92         result_current->exp = p1->exp;
93         result_current->next = result_head;
94         result_temp->next = result_current;
95
96         // Increment the result polynomial pointer, and other polynomial
pointers.
97         result_temp = result_temp->next;
98         p1 = p1->next;
99         p2 = p2->next;
100     }
101
102     // If the exponent of the first polynomial is greater than the second one,
add the node to the result polynomial.
103     else if (p1->exp > p2->exp)
104     {
105         result_current = (struct Node *)malloc(sizeof(struct Node));
106         result_current->coeff = p1->coeff;
107         result_current->exp = p1->exp;
108         result_current->next = result_head;
109         result_temp->next = result_current;
110
111         // increment the result polynomial pointer, and p1
112         result_temp = result_temp->next;
113         p1 = p1->next;
114     }
115
116     // If the exponent of the second polynomial is greater than the first one,
add the node to the result polynomial.
117     else if (p2->exp > p1->exp)
118     {
119         result_current = (struct Node *)malloc(sizeof(struct Node));
120         result_current->coeff = p2->coeff;
121         result_current->exp = p2->exp;
122         result_current->next = result_head;
123         result_temp->next = result_current;
124
125         // increment the result polynomial pointer, and p2
126         result_temp = result_temp->next;
127         p2 = p2->next;
```

```
128     }
129 }
130
131 // Case when p2 exhausts before p1.
132 if (p1 == head1 && p2 != head2)
133 {
134     result_temp->next = p2;
135
136     // This loop is to make the last node of the result polynomial point to
137     the head of the result polynomial.
138     while (result_temp->next != head2)
139     {
140         result_temp = result_temp->next;
141     }
142     result_temp->next = result_head;
143 }
144
145 // Case when p1 exhausts before p2.
146 else if (p1 != head1 && p2 == head2)
147 {
148     result_temp->next = p1;
149     while (result_temp->next != head1)
150     {
151         result_temp = result_temp->next;
152     }
153     result_temp->next = result_head;
154 }
155
156 // Case when both p1 and p2 exhaust.
157 else if (p1 != head1 && p2 != head2)
158 {
159     result_temp->next = p1;
160     while (result_temp != head1)
161     {
162         result_temp = result_temp->next;
163     }
164     result_temp->next = result_head;
165
166     result_temp->next = p2;
167     while (result_temp != head2)
168     {
169         result_temp = result_temp->next;
170     }
171     result_temp->next = result_head;
172 }
173
174 return result_head;
175 }
176
177 int main()
178 {
179     int choice = 0;
180     printf("Hello! What do you want to do? \n Remember to enter linked list in the
181     descending order of exponents. \n");
182     struct Node *head = (struct Node *)malloc(sizeof(struct Node));
183     struct Node *head2 = (struct Node *)malloc(sizeof(struct Node));
184     struct Node *head3 = (struct Node *)malloc(sizeof(struct Node));
185     struct Node *added;
```

```
185     while (choice != 4)
186     {
187         printf("\n\
188 1. Create a Polynomial to represent it in a circular linked list. \n\
189 2. Add 2 Polynomials\n\
190 3. Display your Polynomial\n\
191 4. Quit\n");
192         scanf("%d", &choice);
193         switch (choice)
194         {
195             case 1:
196                 accept_polynomial(head);
197                 display_polynomial(head);
198                 break;
199             case 2:
200                 printf("Please enter the first polynomial");
201                 accept_polynomial(head2);
202                 printf("\nThe First Polynomial you entered is: \n");
203                 display_polynomial(head2);
204
205                 printf("Please enter the second polynomial");
206                 accept_polynomial(head3);
207                 printf("\nThe Second Polynomial you entered is: \n");
208                 display_polynomial(head3);
209
210                 printf("\nThe Added Polynomial: \n");
211                 added = add_polynomials(head2, head3);
212                 display_polynomial(added);
213                 break;
214             case 3:
215                 display_polynomial(head);
216             case 4:
217                 break;
218             default:
219                 printf("\nInvalid\n");
220                 break;
221         }
222     }
223
224     return 0;
225 }
```

## 9.2 Input and Output

```
1 Hello! What do you want to do?
2 Remember to enter linked list in the descending order of exponents.
3
4 1. Create a Polynomial to represent it in a circular linked list.
5 2. Add 2 Polynomials
6 3. Display your Polynomial
7 4. Quit
8 1
9
10 Enter the Coefficient: 1
11
12 Enter the Exponent: 1
13 Do you want to enter more terms? (0 for no, 1 for yes)
14 1
15
```

```
16 Enter the Coefficient: 2
17
18 Enter the Exponent: 3
19 Do you want to enter more terms? (0 for no, 1 for yes)
20 0
21  $1x^1 + 2x^3$ 
22
23 1. Create a Polynomial to represent it in a circular linked list.
24 2. Add 2 Polynomials
25 3. Display your Polynomial
26 4. Quit
27 2
28 Please enter the first polynomial
29 Enter the Coefficient: 1
30
31 Enter the Exponent: 3
32 Do you want to enter more terms? (0 for no, 1 for yes)
33 1
34
35 Enter the Coefficient: 2
36
37 Enter the Exponent: 2
38 Do you want to enter more terms? (0 for no, 1 for yes)
39 1
40
41 Enter the Coefficient: 1
42
43 Enter the Exponent: 1
44 Do you want to enter more terms? (0 for no, 1 for yes)
45 0
46
47 The First Polynomial you entered is:
48  $1x^3 + 2x^2 + 1x^1$ 
49 Please enter the second polynomial
50 Enter the Coefficient: 2
51
52 Enter the Exponent: 3
53 Do you want to enter more terms? (0 for no, 1 for yes)
54 1
55
56 Enter the Coefficient: 5
57
58 Enter the Exponent: 2
59 Do you want to enter more terms? (0 for no, 1 for yes)
60 1
61
62 Enter the Coefficient: 1
63
64 Enter the Exponent: 1
65 Do you want to enter more terms? (0 for no, 1 for yes)
66 0
67
68 The Second Polynomial you entered is:
69  $2x^3 + 5x^2 + 1x^1$ 
70
71 The Added Polynomial:
72  $3x^3 + 7x^2 + 2x^1$ 
73
74 1. Create a Polynomial to represent it in a circular linked list.
```



```
75 2. Add 2 Polynomials
76 3. Display your Polynomial
77 4. Quit
78 4
```

## **10 Conclusion**

## **11 FAQ**