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

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

```
// Circular linked List
// Implementing the following polynomial operations using circular linked list.
    Create Dislay and Add.

// Krishnaraj Thadesar
// Jan 22nd 2023
// Assignment 1
// Advanced Data Structures
// Semester 4

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

// The default way to represent circular linked lists using structures.
struct Node
```

```
{
    int coeff;
    int exp;
    struct Node *next;
};

// Function to accept the polynomial from the user.
void accept_polynomial(struct Node *head)
{
    struct Node *temp = head;
    int choice = 0;
    do
    {
        struct Node *curr = (struct Node *)malloc(sizeof(struct Node));
        printf("\nEnter the Coefficient: ");
        scanf("%d", &curr->coeff);
        printf("\nEnter the Exponent: ");
        scanf("%d", &curr->exp);

        // Main Logic of inserting node at the end and making it point to the head
    .
        curr->next = head;
        temp->next = curr;
        temp = temp->next;

        printf("Do you want to enter more terms? (0 for no, 1 for yes) \n");
        scanf("%d", &choice);
    } while (choice != 0);
}

// Function to display the polynomial.
int display_polynomial(struct Node *head)
{
    // Edge Case of empty list.
    if (head->next == head)
    {
        printf("\nNo terms in the polynomial");
        return -1;
    }

    struct Node *curr = (struct Node *)malloc(sizeof(struct Node));
    curr = head->next;

    while (curr != head)
    {
        printf("%dx^%d", curr->coeff, curr->exp);
        curr = curr->next;
        if (curr != head)
        {
            printf(" + ");
        }
    }
    printf("\n");
}

// Function to add two polynomials, Returns the head of the added polynomial.
// Takes as input the heads of the other 2 polynomials that you want to add.
struct Node *add_polynomials(struct Node *head1, struct Node *head2)
{
```

```
    // Pointers for the result polynomial.
    struct Node *result_head = (struct Node *)malloc(sizeof(struct Node));
    result_head->next = result_head;
    struct Node *result_temp = result_head;
    struct Node *result_current;

    // p1 and p2 are the pointers to the first node of the two polynomials.
    struct Node *p1 = head1->next;
    struct Node *p2 = head2->next;

    // In case one of the polynomial exhausts before the other one.
    while (p1 != head1 && p2 != head2)
    {
        // if the exponents are equal, add the coefficients and add the node to
    the result polynomial.
        if (p1->exp == p2->exp)
        {
            // Copy the data of thesum of the nodes to the result polynomial.
            result_current = (struct Node *)malloc(sizeof(struct Node));
            result_current->coeff = p1->coeff + p2->coeff;
            result_current->exp = p1->exp;
            result_current->next = result_head;
            result_temp->next = result_current;

            // Increment the result polynomial pointer, and other polynomial
    pointers.
            result_temp = result_temp->next;
            p1 = p1->next;
            p2 = p2->next;
        }

        // If the exponent of the first polynomial is greater than the second one,
     add the node to the result polynomial.
        else if (p1->exp > p2->exp)
        {
            result_current = (struct Node *)malloc(sizeof(struct Node));
            result_current->coeff = p1->coeff;
            result_current->exp = p1->exp;
            result_current->next = result_head;
            result_temp->next = result_current;

            // increment the result polynomial pointer, and p1
            result_temp = result_temp->next;
            p1 = p1->next;
        }

        // If the exponent of the second polynomial is greater than the first one,
     add the node to the result polynomial.
        else if (p2->exp > p1->exp)
        {
            result_current = (struct Node *)malloc(sizeof(struct Node));
            result_current->coeff = p2->coeff;
            result_current->exp = p2->exp;
            result_current->next = result_head;
            result_temp->next = result_current;

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

```
    while (choice != 4)
    {
        printf("\n\
1. Create a Polynomial to represent it in a circular linked list. \n\
2. Add 2 Polynomials\n\
3. Display your Polynomial\n\
4. Quit\n");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            accept_polynomial(head);
            display_polynomial(head);
            break;
        case 2:
            printf("Please enter the first polynomial");
            accept_polynomial(head2);
            printf("\nThe First Polynomial you entered is:  \n");
            display_polynomial(head2);

            printf("Please enter the second polynomial");
            accept_polynomial(head3);
            printf("\nThe Second Polynomial you entered is:  \n");
            display_polynomial(head3);

            printf("\nThe Added Polynomial:  \n");
            added = add_polynomials(head2, head3);
            display_polynomial(added);
            break;
        case 3:
            display_polynomial(head);
        case 4:
            break;
        default:
            printf("\nInvalid\n");
            break;
        }
    }

    return 0;
}
```

9.2 Input and Output

```
Hello! What do you want to do?
 Remember to enter linked list in the descending order of exponents.

1. Create a Polynomial to represent it in a circular linked list.
2. Add 2 Polynomials
3. Display your Polynomial
4. Quit
1

Enter the Coefficient: 1

Enter the Exponent: 1
Do you want to enter more terms? (0 for no, 1 for yes)
1

```

```
Enter the Coefficient: 2

Enter the Exponent: 3
Do you want to enter more terms? (0 for no, 1 for yes)
0
1x^1 + 2x^3

1. Create a Polynomial to represent it in a circular linked list.
2. Add 2 Polynomials
3. Display your Polynomial
4. Quit
2
Please enter the first polynomial
Enter the Coefficient: 1

Enter the Exponent: 3
Do you want to enter more terms? (0 for no, 1 for yes)
1

Enter the Coefficient: 2

Enter the Exponent: 2
Do you want to enter more terms? (0 for no, 1 for yes)
1

Enter the Coefficient: 1

Enter the Exponent: 1
Do you want to enter more terms? (0 for no, 1 for yes)
0

The First Polynomial you entered is:
1x^3 + 2x^2 + 1x^1
Please enter the second polynomial
Enter the Coefficient: 2

Enter the Exponent: 3
Do you want to enter more terms? (0 for no, 1 for yes)
1

Enter the Coefficient: 5

Enter the Exponent: 2
Do you want to enter more terms? (0 for no, 1 for yes)
1

Enter the Coefficient: 1

Enter the Exponent: 1
Do you want to enter more terms? (0 for no, 1 for yes)
0

The Second Polynomial you entered is:
2x^3 + 5x^2 + 1x^1

The Added Polynomial:
3x^3 + 7x^2 + 2x^1

1. Create a Polynomial to represent it in a circular linked list.
```

```
2. Add 2 Polynomials
3. Display your Polynomial
4. Quit
4
```

10 Conclusion

11 FAQ