**Polynomial Linked List**

Problem Statement:

Write a menu driven program to do the following:

1. Create a polynomial expression of the form AXp + Bxq- C xr ..... ..... .. Let it be the current polynomial.

2. Add another polynomial to the current polynomial

3. Multiply another polynomial to the current polynomial

4. Display a polynomial

5. Delete a polynomial

6. Delete all linked lists in the memory and exit

Note: with avail list maintenance Deletion (option 5) should take only constant time.

• Output example :

Type 1 to create a polynomial

Type 2 to add another polynomial to the current polynomial

Type 3 to multiply another polynomial to the current polynomial

Type 4 to display a polynomial

Type 5 to delete a polynomial

Type 6 to delete all linked lists in the memory and exit

Please give option between 1 to 6

1

Enter the number of terms: 3

Enter the cofficient for term 1: 1

Enter the exponent for term 1: 2

Enter the cofficient for term 2: 3

Enter the exponent for term 2: 4

Enter the cofficient for term 3: 5

Enter the exponent for term 3: 6

4

5x^6 + 3x^4 + 1x^2

2

Enter the second polynomial

Enter the number of terms: 5

Enter the cofficient for term 1: 8

Enter the exponent for term 1: 2

Enter the cofficient for term 2: 1

Enter the exponent for term 2: 2

Enter the cofficient for term 3: 3

Enter the exponent for term 3: 6

Enter the cofficient for term 4: 5

Enter the exponent for term 4: 7

Enter the cofficient for term 5: 9

Enter the exponent for term 5: 1

4

5x^7 + 8x^6 + 3x^4 + 10x^2 + 9x^1

3

Enter the second polynomial

Enter the number of terms: 2

Enter the cofficient for term 1: 5

Enter the exponent for term 1: 4

Enter the cofficient for term 2: 2

Enter the exponent for term 2: 1

4

25x^14 + 55x^13 + 24x^12 + 40x^11 + 49x^10 + 95x^9 + 217x^8 + 115x^7 + 77x^6 + 78x^5 + 90x^4 + 191x^3 + 99x^2

5

Polynomial Deleted

6

Terminated

Proposed C Code:

/\* ------- main.c ------- \*/

#include <stdio.h>

#include <stdlib.h>

typedef struct N

{

int coefficient;

int exponent;

struct N \*next;

} Node;

Node \*avail = NULL; //Initializing the avail list

Node \*delete (Node \*head) //Delete polynomial

{

Node \*temp;

if (temp != NULL)

{

temp = head->next;

head->next = avail;

avail = temp;

head = NULL;

}

return head;

}

Node \*insert(Node \*head, int co, int ex) //Insert a node in the polynomial

{

Node \*temp;

Node \*newP = (Node \*)malloc(sizeof(Node));

newP->coefficient = co;

newP->exponent = ex;

newP->next = NULL;

if (head == NULL)

{

head = newP;

head->next = head;

}

else if (ex > head->exponent)

{

newP->exponent = head->exponent;

newP->coefficient = head->coefficient;

head->exponent = ex;

head->coefficient = co;

newP->next = head->next;

head->next = newP;

}

else

{

if (head->exponent == ex)

{

head->coefficient = head->coefficient + co;

}

else

{

temp = head;

while (temp->next != head && temp->next->exponent >= ex)

{

temp = temp->next;

}

if (temp->exponent == ex)

{

temp->coefficient = temp->coefficient + co;

}

else

{

newP->next = temp->next;

temp->next = newP;

}

}

}

return head;

}

Node \*create(Node \*head) // Creating a polynomial

{

int n;

int coeff;

int expo;

printf("Enter the number of terms: ");

scanf("%d", &n);

for (int i = 0; i < n; i++)

{

printf("Enter the cofficient for term %d: ", i + 1);

scanf("%d", &coeff);

printf("Enter the exponent for term %d: ", i + 1);

scanf("%d", &expo);

head = insert(head, coeff, expo);

}

return head;

}

Node \*add(Node \*head, Node \*shead) // Addition of two polynomial

{

Node \*temp1 = head;

Node \*temp2 = shead;

Node \*addition = NULL;

if (head == NULL)

{

head = shead;

}

else

{

do

{

head = insert(head, temp2->coefficient, temp2->exponent);

temp2 = temp2->next;

} while (temp2 != shead);

}

return head;

}

Node \*multiply(Node \*head, Node \*shead) //Multiplication of two polynomial

{

Node \*temp1 = head;

Node \*temp2 = shead;

Node \*multiplication = NULL;

if (head == NULL)

return shead;

do

{

do

{

multiplication = insert(multiplication, temp1->coefficient \* temp2->coefficient, temp1->exponent + temp2->exponent);

temp2 = temp2->next;

} while (temp2 != shead);

temp1 = temp1->next;

} while (temp1 != head);

head = multiplication;

return head;

}

Node \*deleteAll(Node \*avail) // Delete all lists with the avail list

{

if (avail == NULL)

{

printf("Avail list is already empty\n");

}

else

{

Node \*temp = avail;

while (avail != NULL)

{

avail = avail->next;

}

free(temp);

}

return avail;

}

void display(Node \*head) // Display the polynomial

{

if (head == NULL)

{

printf("No Polynomial\n");

}

else

{

Node \*temp = head;

do

{

printf("%dx^%d", temp->coefficient, temp->exponent);

temp = temp->next;

if (temp != head)

{

printf(" + ");

}

} while (temp != head);

}

printf("\n");

}

int main()

{

Node \*head = NULL;

Node \*shead = NULL;

int n;

printf("Type 1 to create a polynomial\nType 2 to add another polynomial to the current polynomial \nType 3 to multiply another polynomial to the current polynomial \nType 4 to display a polynomial \nType 5 to delete a polynomial \nType 6 to delete all linked lists in the memory and exit\n\n");

printf("Please give option between 1 to 6\n");

do

{

scanf("%d", &n);

switch (n)

{

case 1:

head = create(head);

break;

case 2:

printf("Enter the second polynomial\n");

shead = create(shead);

head = add(head, shead);

break;

case 3:

printf("Enter the second polynomial\n");

shead = create(shead);

head = multiply(head, shead);

break;

case 4:

display(head);

break;

case 5:

head = delete (head);

printf("Polynomial Deleted\n");

break;

case 6:

printf("Terminated\n");

head = delete (head);

avail = deleteAll(avail);

break;

}

} while (n != 6);

return 0;

}

/\* ---------------------- \*/

Conclusion:

The proposed algorithm has a runtime of O(nxm), where n and m are the lengths of the two linked lists.

Limitations and assumptions for this algorithm include:

1.Thought here deletion takes place in O(1) time but insertion takes place in O(n) in stead of O(1).

2.Here multiplication takes place in O(n2) time.

3.In menu selection user should not give any input except 1 to 6.