LAB-07

Name: K V Jaya Harsha

Roll no: CS23B1034 Date: 22-10-2024

Q1. Multiply two polynomial expressions by considering three variables x,y, and z using a linked list. (in cpp)

polynomial.cpp > main()

```cpp
// CS23B1034
// K V Jaya Harsha
#include <iostream>
using namespace std;
struct Node{
    int coeff;
    int powX;
    int powY;
    int powZ;
    Node *next;
};
Node *createNode(int coeff, int powX, int powY, int powZ, Node *head){
    Node *newNode = new Node;
    newNode->coeff = coeff;
    newNode->powX = powX;
    newNode->powY = powY;
    newNode->powZ = powZ;
    newNode->next = nullptr;
    if (head == nullptr){
        return newNode;
    }
    Node *temp = head;
    while (temp->next != nullptr){
        temp = temp->next;
    }
    temp->next = newNode;
    return head;
}
Node *addOrUpdateNode(Node *head, int coeff, int powX, int powY, int powZ){
    Node *temp = head;
    Node *prev = nullptr;
    while (temp != nullptr){
        if (temp->powX == powX && temp->powY == powY && temp->powZ == powZ){
            temp->coeff += coeff;
            return head;
        }
        prev = temp;
        temp = temp->next;
    }
    Node *newNode = new Node;
    newNode->coeff = coeff;
    newNode->powX = powX;
    newNode->powY = powY;
    newNode->powZ = powZ;
    newNode->next = nullptr;
    if (prev == nullptr){
        return newNode;
    }
    else{
        prev->next = newNode;
    }
    return head;
}
void displayPoly(Node *head){
```

```
}
void displayPoly(Node *head){
    Node *temp = head;
    bool first = true;
    while (temp != nullptr){
        if (temp->coeff != 0){
            if (!first && temp->coeff > 0){
                cout << " + ";
            }
            cout << temp->coeff << "x^" << temp->powX << "y^" << temp->powY << "z^" << temp->powZ;
            first = false;
        }
        temp = temp->next;
    }
    cout << endl;
}
Node *multiplyPolynomials(Node *poly1, Node *poly2){
    Node *result = nullptr;
    for (Node *temp1 = poly1; temp1 != nullptr; temp1 = temp1->next){
        for (Node *temp2 = poly2; temp2 != nullptr; temp2 = temp2->next){
            int coeff = temp1->coeff * temp2->coeff;
            int powX = temp1->powX + temp2->powX;
            int powY = temp1->powY + temp2->powY;
            int powZ = temp1->powZ + temp2->powZ;
            result = addOrUpdateNode(result, coeff, powX, powY, powZ);
        }
    }
    return result;
}
int main(){
    Node *poly1 = nullptr;
    poly1 = createNode(3, 2, 1, 0, poly1);
    poly1 = createNode(2, 1, 1, 1, poly1);
    poly1 = createNode(5, 0, 0, 0, poly1);
    Node *poly2 = nullptr;
    poly2 = createNode(1, 2, 2, 0, poly2);
    poly2 = createNode(1, 1, 1, 1, poly2);
    poly2 = createNode(1, 0, 0, 0, poly2);

    cout << "First Polynomial: ";
    displayPoly(poly1);
    cout << "Second Polynomial: ";
    displayPoly(poly2);

    Node *result = multiplyPolynomials(poly1, poly2);
    cout << "Result of Multiplication: ";
    displayPoly(result);

    return 0;
}
```

```
First Polynomial: 3x^2y^1z^0 + 2x^1y^1z^1 + 5x^0y^0z^0
Second Polynomial: 1x^2y^2z^0 + 1x^1y^1z^1 + 1x^0y^0z^0
Result of Multiplication: 3x^4y^3z^0 + 3x^3y^2z^1 + 3x^2y^1z^0 + 2x^3y^3z^1 + 2x^2y^2z^2 + 7x^1y^1z^1 + 5x^2y^2z^0 + 5x^0y^0z^0
```

Q2. Implement priority queue. (in cpp)

```cpp
//CS23B1034
//K V Jaya Harsha
#include <iostream>
using namespace std;
struct Node{
    int coeff;
    int powX;
    int powY;
    int powZ;
    Node *next;
};
Node *createNode(int coeff, int powX, int powY, int powZ){
    Node *newNode = new Node;
    newNode->coeff = coeff;
    newNode->powX = powX;
    newNode->powY = powY;
    newNode->powZ = powZ;
    newNode->next = nullptr;
    return newNode;
}
Node *insertWithPriority(Node *head, int coeff, int powX, int powY, int powZ){
    Node *newNode = createNode(coeff, powX, powY, powZ);
    if (head == nullptr || (newNode->powX > head->powX) ||
        (newNode->powX == head->powX && newNode->powY > head->powY) ||
        (newNode->powX == head->powX && newNode->powY == head->powY && newNode->powZ > head->powZ)){
        newNode->next = head;
        return newNode;
    }
    Node *temp = head;
    while (temp->next != nullptr &&
           ((temp->next->powX > newNode->powX) ||
           (temp->next->powX == newNode->powX && temp->next->powY > newNode->powY) ||
           (temp->next->powX == newNode->powX && temp->next->powY == newNode->powY && temp->next->powZ > newNode->powZ))){
        temp = temp->next;
    }
    newNode->next = temp->next;
    temp->next = newNode;
    return head;
}
```

```
}
Node *addOrUpdateNode(Node *head, int coeff, int powX, int powY, int powZ){
    Node *temp = head;
    while (temp != nullptr)    {
        if (temp->powX == powX && temp->powY == powY && temp->powZ == powZ){
            temp->coeff += coeff;
            return head;
        }
        temp = temp->next;
    }
    return insertWithPriority(head, coeff, powX, powY, powZ);
}
void displayPoly(Node *head){
    Node *temp = head;
    bool first = true;
    while (temp != nullptr){
        if (temp->coeff != 0){
            if (!first && temp->coeff > 0){
                cout << " + ";
            }
            cout << temp->coeff << "x^" << temp->powX << "y^" << temp->powY << "z^" << temp->powZ;
            first = false;
        }
        temp = temp->next;
    }
    cout << endl;
}
Node *multiplyPolynomials(Node *poly1, Node *poly2){
    Node *result = nullptr;
    for (Node *temp1 = poly1; temp1 != nullptr; temp1 = temp1->next){
        for (Node *temp2 = poly2; temp2 != nullptr; temp2 = temp2->next){
            int coeff = temp1->coeff * temp2->coeff;
            int powX = temp1->powX + temp2->powX;
            int powY = temp1->powY + temp2->powY;
            int powZ = temp1->powZ + temp2->powZ;
            result = addOrUpdateNode(result, coeff, powX, powY, powZ);
        }
    }
    return result;
}
int main(){
```

```cpp
int main(){
    Node *poly1 = nullptr;
    poly1 = insertWithPriority(poly1, 3, 2, 4, 0);
    poly1 = insertWithPriority(poly1, 2, 1, 1, 1);
    poly1 = insertWithPriority(poly1, 5, 0, 0, 0);
    Node *poly2 = nullptr;
    poly2 = insertWithPriority(poly2, 1, 2, 2, 0);
    poly2 = insertWithPriority(poly2, 1, 1, 1, 1);
    poly2 = insertWithPriority(poly2, 1, 0, 0, 0);

    cout << "First Polynomial: ";
    displayPoly(poly1);
    cout << "Second Polynomial: ";
    displayPoly(poly2);

    Node *result = multiplyPolynomials(poly1, poly2);
    cout << "Result: ";
    displayPoly(result);

    return 0;
}
```

```
labs\lab-7\" ; if ($?) { g++ polynomial-priority.cpp -o polynomial-priority } ; if ($?) { .\polynomial-priority
First Polynomial: 3x^2y^4z^0 + 2x^1y^1z^1 + 5x^0y^0z^0
Second Polynomial: 1x^2y^2z^0 + 1x^1y^1z^1 + 1x^0y^0z^0
Result: 3x^4y^6z^0 + 3x^3y^5z^1 + 2x^3y^3z^1 + 3x^2y^4z^0 + 2x^2y^2z^2 + 5x^2y^2z^0 + 7x^1y^1z^1 + 5x^0y^0z^0
PS C:\Users\harsh\OneDrive\Documents\Desktop\challenge\dsa\labs\lab-7>
```