/\*

Implement a polynomial class, with following properties and functions.

Properties :

1. An integer (lets say A) which holds the coefficient and degrees. Use array indices as degree and A[i] as coefficient of ith degree.

2. An integer holding total size of array A.

Functions :

1. Default constructor

2. Copy constructor

3. setCoefficient -

This function sets coefficient for a particular degree value. If the given degree is greater than the current capacity of polynomial, increase the capacity accordingly and add then set the required coefficient. If the degree is within limits, then previous coefficient value is replaced by given coefficient value

4. Overload "+" operator (P3 = P1 + P2) :

Adds two polynomials and returns a new polynomial which has result.

5. Overload "-" operator (P3 = p1 - p2) :

Subtracts two polynomials and returns a new polynomial which has result

6. Overload \* operator (P3 = P1 \* P2) :

Multiplies two polynomials and returns a new polynomial which has result

7. Overload "=" operator (Copy assignment operator) -

Assigns all values of one polynomial to other.

8. print() -

Prints all the terms (only terms with non zero coefficients are to be printed) in increasing order of degree.

Print pattern for a single term : <coefficient>"x"<degree>

And multiple terms should be printed separated by space. And after printing one polynomial, print new line. For more clarity, refer sample test cases

Input Format :

Line 1 : N, total number of terms in polynomial P1

Line 2 : N integers representing degree of P1 (separated by space)

Line 3 : N integers representing coefficients of P1 (separated by space)

Line 4 : M, total number of terms in polynomial P2

Line 5 : M integers representing degree of P2 (separated by space)

Line 6 : M integers representing coefficients of P2 (separated by space)

Line 7 : Integer C, choice representing the function to be called (See main for more details)

Sample Input 1 :

3

1 3 5

1 2 -4

4

0 1 2 3

4 2 -3 1

1

Sample Output 1 :

4x0 3x1 -3x2 3x3 -4x5

\*/

ANSWER:-

class Polynomial {

public:

int \*degCoeff; // Name of your array (Don't change this)

int capacity;

Polynomial(){

this->degCoeff=new int[6];

this->capacity=5;

}

//Making paramaterized constructor for defining with capacity

Polynomial (int capacity){

this->degCoeff=new int[capacity+1];

this->capacity=capacity;

}

Polynomial (Polynomial const &p){

int \*newdeg=new int[p.capacity+1];

for(int i=0;i<=p.capacity;i++)

newdeg[i]=p.degCoeff[i];

this->degCoeff=newdeg;

this->capacity=p.capacity;

}

void setCoefficient(int deg,int coef){

if(deg>capacity){

int newcapacity=deg;

int \*newdeg=new int[newcapacity+1];

//Copy the contents from original to new

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

newdeg[i]=degCoeff[i];

this->degCoeff=newdeg;

this->capacity=newcapacity;

//Set the new coeff;

degCoeff[deg]=coef;

}

else{

degCoeff[deg]=coef;

}

}

Polynomial operator+(Polynomial const &P2){

int newcap=max(this->capacity,P2.capacity);

Polynomial P3(newcap);

for(int i=0;i<=newcap;i++){

if(i<=capacity && i<=P2.capacity)

P3.degCoeff[i]=this->degCoeff[i]+P2.degCoeff[i];

else if(i<=capacity)

P3.degCoeff[i]=this->degCoeff[i];

else

P3.degCoeff[i]=P2.degCoeff[i];

}

return P3;

}

Polynomial operator-(Polynomial const &P2){

int newcap=max(this->capacity,P2.capacity);

Polynomial P3(newcap);

for(int i=0;i<=newcap;i++){

if(i<=capacity && i<=P2.capacity)

P3.degCoeff[i]=this->degCoeff[i]-P2.degCoeff[i];

else if(i<=capacity)

P3.degCoeff[i]=this->degCoeff[i];

else

P3.degCoeff[i]=-P2.degCoeff[i];

}

return P3;

}

Polynomial operator\*(Polynomial const &P2){

int newcap=this->capacity+P2.capacity;

Polynomial P3(newcap);

for(int i=0;i<=this->capacity;i++){

for(int j=0;j<=P2.capacity;j++){

P3.degCoeff[i+j]+=this->degCoeff[i]\*P2.degCoeff[j];

}

}

return P3;

}

void operator=(Polynomial const &p){

int \*newdeg=new int[p.capacity+1];

//Copy the contents

for(int i=0;i<p.capacity;i++)

newdeg[i]=p.degCoeff[i];

this->degCoeff=newdeg;

this->capacity=p.capacity;

}

void print(){

for(int i=0;i<=this->capacity;i++){

if(degCoeff[i]!=0)

cout<<degCoeff[i]<<"x"<<i<<" ";

}

cout<<endl;

}

};

/\* C++ implementation to convert infix expression to postfix\*/

// Note that here we use std::stack for Stack operations

#include <vector>

#include <climits>

#include <iostream>

using namespace std;

#include "Solution.h"

//Driver program to test above functions

int main()

{

int count1,count2,choice;

cin >> count1;

int \*degree1 = new int[count1];

int \*coeff1 = new int[count1];

for(int i=0;i < count1; i++) {

cin >> degree1[i];

}

for(int i=0;i < count1; i++) {

cin >> coeff1[i];

}

Polynomial first;

for(int i = 0; i < count1; i++){

first.setCoefficient(degree1[i],coeff1[i]);

}

cin >> count2;

int \*degree2 = new int[count2];

int \*coeff2 = new int[count2];

for(int i=0;i < count2; i++) {

cin >> degree2[i];

}

for(int i=0;i < count2; i++) {

cin >> coeff2[i];

}

Polynomial second;

for(int i = 0; i < count2; i++){

second.setCoefficient(degree2[i],coeff2[i]);

}

cin >> choice;

switch(choice){

// Add

case 1:

{

Polynomial result1 = first + second;

result1.print();

break;

}

// Subtract

case 2 :

{

Polynomial result2 = first - second;

result2.print();

break;

}

// Multiply

case 3 :

{

Polynomial result3 = first \* second;

result3.print();

break;

}

case 4 : // Copy constructor

{

Polynomial third(first);

if(third.degCoeff == first.degCoeff) {

cout << "false" << endl;

}

else {

cout << "true" << endl;

}

break;

}

case 5 : // Copy assignment operator

{

Polynomial fourth(first);

if(fourth.degCoeff == first.degCoeff) {

cout << "false" << endl;

}

else {

cout << "true" << endl;

}

break;

}

}

return 0;

}