Q1:

Code:

#include <iostream>

using namespace std;

class RationalNumber

{

private:

int numerator; int denominator;

public:

RationalNumber()

{ }

~RationalNumber() { }

RationalNumber(int n, int d)

{

int max = 0;

if (n > d)

{

max = n;

}

else

{

max = d;

}

for (int i = 2; i <= max / 2; i++)

{

if (n % i == 0 && d % i == 0)

{

n /= i;

d /= i;

}

}

numerator = n;

denominator = d;

cout << "Reduced form =" << endl;

cout << numerator << '/' << denominator << endl;

} RationalNumber operator+(RationalNumber a)

{

int max;

RationalNumber t;

t.numerator = a.numerator \* denominator + a.denominator \* numerator;

t.denominator = a.denominator \* denominator;

if (t.numerator > t.denominator)

{

max = t.numerator;

}

else

{

max = t.denominator;

}

for (int i = 2; i <= max / 2; i++)

{

if (t.numerator % i == 0 && t.denominator % i == 0)

{

t.numerator /= i;

t.denominator /= i;

}

}

return t;

}

RationalNumber operator-(RationalNumber b)

{

RationalNumber t;

t.numerator = b.denominator \* numerator - denominator \* b.numerator;

t.denominator = b.denominator \* denominator;

return t;

}

RationalNumber operator\*(RationalNumber c)

{

RationalNumber t;

t.numerator = c.numerator \* numerator;

t.denominator = c.denominator \* denominator;

return t;

}

RationalNumber operator/(RationalNumber d)

{

RationalNumber t;

t.numerator = d.denominator \* numerator;

t.denominator = denominator \* d.numerator;

return t;

}

void print()

{

cout << numerator << '/' << denominator << endl;

}

};

int main() {

char choice; int n = 0, d = 0;

RationalNumber r1, r2, r3;

do

{

cout << "Enter Positive Values Numerator Denominator" << endl;

cin >> n >> d;

} while ((n <= 0) && (d <= 0));

r1 = RationalNumber(n, d);

do

{

cout << "Enter Positive values for second Numerator & Denominator)" << endl;

cin >> n >> d;

} while ((n <= 0) && (d <= 0));

r2 = RationalNumber(n, d);

do

{

cout << "Enter '+' for Addition '-' for Subtraction '\*' for multiplication '/' for division and 'E' for Exit " << endl;

cin >> choice;

if (choice == '+')

{

r3 = r2 + r1;

}

else if

(choice == '-')

{

r3 = r2 - r1;

}

else if (choice == '\*')

{

r3 = r2 \* r1;

}

else if (choice == '/')

{

r3 = r2 / r1;

}

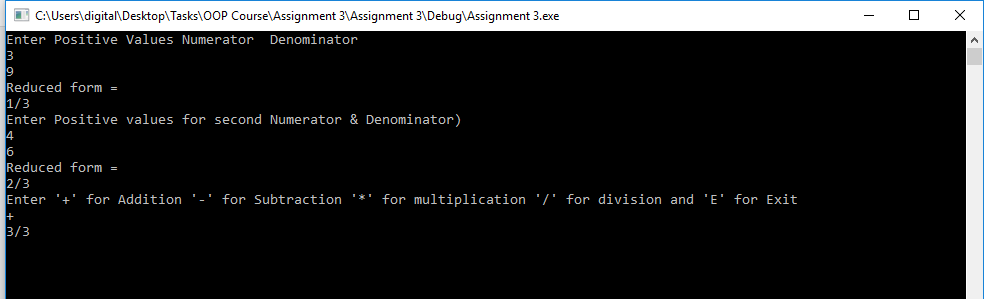
r3.print();

} while (choice == 'E' || choice == 'e');

system("pause>0");

return 0;

}



Q2:

Code:

#include <iostream>

using namespace std;

class Polynomial

{

private:

int \* ptr;

int n;

public:

void create(int n)

{

ptr = new int[n];

}

void set()

{

do

{

cout << "Enter the Degree of Polynomial: ";

cin >> n;

} while (n <= 0);

ptr = new int[n];

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

{

cout << "Enter Coefficients of Polynomials: x^" << n - i << " ";

cin >> ptr[i];

}

}

void get()

{

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

{

if (i != (n - 1))

{

cout << " ( " << ptr[i] << "x^" << n - i << ") + ";

}

else

{

cout << " ( " << ptr[i] << "x^" << n - i << ")";

}

}

cout << endl;

}

Polynomial operator+(Polynomial c)

{

int l;

if (n > c.n)

{

l = n;

}

else

{

l = c.n;

}

Polynomial temp;

temp.create(l);

int count = 0, count1 = 0;

while (count == l);

{

if ((n - count) == (c.n - count1))

{

temp.ptr[count] = ptr[count] + c.ptr[count1];

count++;

count1++;

}

else if (n > c.n)

{

count++;

}

else

{

count1++;

}

}

return temp;

}

Polynomial operator-(Polynomial c)

{

int l;

if (n > c.n)

{

l = n;

}

else

{

l = c.n;

}

Polynomial temp;

temp.create(l);

int count = 0, count1 = 0;

while (count == l);

{

if ((n - count) == (c.n - count1))

{

temp.ptr[count] = ptr[count] - c.ptr[count1];

count++;

count1++;

}

else if (n > c.n)

{

count++;

}

else

{

count1++;

}

}

return temp;

}

Polynomial operator\*(Polynomial c)

{

int l;

if (n > c.n)

{

l = n;

}

else

{

l = c.n;

}

Polynomial temp;

temp.create(l);

int count = 0, count1 = 0;

while (count == l);

{

if ((n - count) == (c.n - count1))

{

temp.ptr[count] = ptr[count] \* c.ptr[count1];

count++;

count1++;

}

else if(n>c.n)

{

count++;

}

else

{

count1++;

}

}

return temp;

}

void operator=(Polynomial c)

{

if (n > c.n)

{

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

{

ptr[i + 1] = c.ptr[i];

}

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

{

if (i != (n - 1))

{

cout << " ( " << ptr[i] << "x^" << n - i << ") + ";

}

else

{

cout << " ( " << ptr[i] << "x^" << n - i << ")";

}

}

}

else if (n == c.n)

{

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

{

ptr[i] = c.ptr[i];

}

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

{

if (i != (n - 1))

{

cout << " ( " << ptr[i] << "x^" << n - i << ") + ";

}

else

{

cout << " ( " << ptr[i] << "x^" << n - i << ")";

}

}

}

else

{

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

{

c.ptr[i+1] = ptr[i];

}

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

{

if (i != (n - 1))

{

cout << " ( " << ptr[i] << "x^" << n - i << ") + ";

}

else

{

cout << " ( " << ptr[i] << "x^" << n - i << ")";

}

}

}

}

void operator+=(int n)

{

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

{

ptr[i] = ptr[i] + n;

}

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

{

if (i != (n - 1))

{

cout << " ( " << ptr[i] << "x^" << n - i << ") + ";

}

else

{

cout << " ( " << ptr[i] << "x^" << n - i << ")";

}

}

}

void operator\*=(int n)

{

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

{

ptr[i] = ptr[i] \* n;

}

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

{

if (i != (n - 1))

{

cout << " ( " << ptr[i] << "x^" << n - i << ") + ";

}

else

{

cout << " ( " << ptr[i] << "x^" << n - i << ")";

}

}

}

void operator-=(int n)

{

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

{

ptr[i] = ptr[i] - n;

}

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

{

if (i != (n - 1))

{

cout << " ( " << ptr[i] << "x^" << n - i << ") + ";

}

else

{

cout << " ( " << ptr[i] << "x^" << n - i << ")";

}

}

}

};

int main()

{

Polynomial m1, m2;

m1.set();

m2.set();

m1.get();

m2.get();

Polynomial m3 = m1 + m2;

m3.get();

Polynomial m4 = m1 - m2;

m4.get();

Polynomial m5 = m1 \* m2;

m5.get();

m1 += 1;

m1 -= 1;

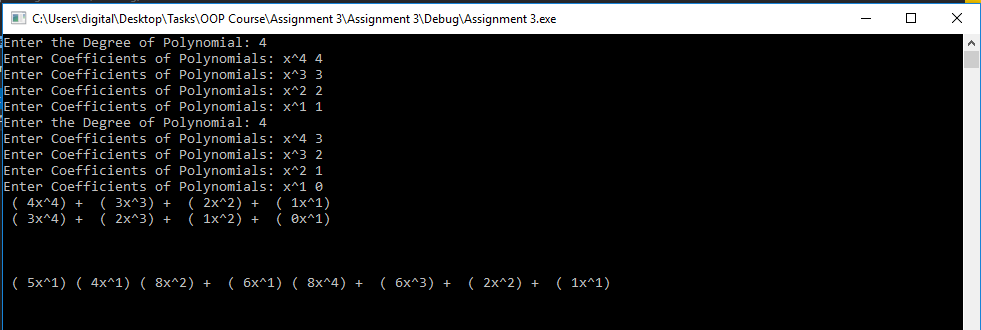
m1 \*= 2;

m1 = m2;

system("pause>0");

return 0;

}



Q3:

Code:

#include<iostream>

using namespace std;

class complex

{

private:

int real, imaginary;

public:

complex()

{

real = 0;

imaginary = 0;

}

complex(int r, int i)

{

real = r;

imaginary = i;

}

complex operator+(complex obj)

{

complex temp;

temp.real = real + obj.real;

temp.imaginary = imaginary + obj.imaginary;

return temp;

}

void display()

{

cout << "Answer= " << real << "+" << imaginary << "i" << endl;

}

complex operator-(complex obj)

{

complex temp;

temp.real = real - obj.real;

temp.imaginary = imaginary - obj.imaginary;

return temp;

}

complex operator\*(complex obj)

{

complex temp;

temp.real = real \* obj.real;

temp.imaginary = imaginary \* obj.imaginary;

return temp;

}

complex operator/(complex obj)

{

complex temp;

temp.real = real / obj.real;

temp.imaginary = imaginary / obj.imaginary;

return temp;

}

};

int main()

{

int a, b, c, d;

cout << "Enter Values two Real Vales: ";

cin >> a >> c;

cout << "Enter Values Real Vales: ";

cin >> b >> d;

complex C1(a, b), C2(c, d), C3;

cout << "+" << endl;

C3 = C2 + C1;

C3.display();

cout << "-" << endl;

C3 = C2 - C1;

C3.display();

cout << "\*" << endl;

C3 = C2 \* C1;

C3.display();

cout << "/" << endl;

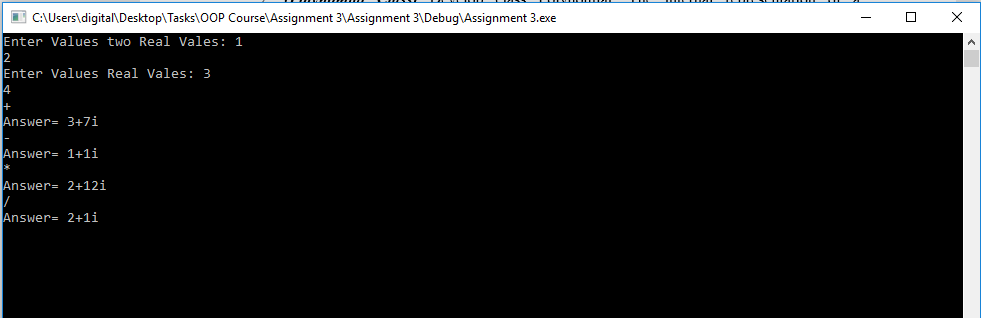
C3 = C2 / C1;

C3.display();

system("pause>0");

return 0;

}



Q4:

Code:

Q5:

Code: