**202211341 컴퓨터공학부 이윤희**

|  |
| --- |
| **Question1** |
| **Source code** |
| #include<iostream>  #include<iomanip>  using namespace std;  class Complex {  private:  double x;  double y;  public:  Complex(double x, double y);  friend const Complex operator+(const Complex& left, const Complex& right);  friend const Complex operator-(const Complex& left, const Complex& right);  friend const Complex operator\*(const Complex& left, const Complex& right);  friend const Complex operator/(const Complex& left, const Complex& right);  friend ostream& operator<<(ostream& left, const Complex& right);  Complex& operator+=(const Complex right);  Complex& operator-=(const Complex right);  Complex& operator\*=(const Complex right);  Complex& operator/=(const Complex right);  };  Complex::Complex(double x, double y) :x(x), y(y) {}  const Complex operator+(const Complex& left, const Complex& right) {  double newX = left.x + right.x;  double newY = left.y + right.y;  Complex result(newX, newY);  return result;  }  const Complex operator-(const Complex& left, const Complex& right) {  double newX = left.x - right.x;  double newY = left.y - right.y;  Complex result(newX, newY);  return result;  }  const Complex operator\*(const Complex& left, const Complex& right) {  double newX = left.x \* right.x - left.y \* right.y;  double newY = left.x \* right.y + right.x \* left.y;  Complex result(newX, newY);  return result;  }  const Complex operator/(const Complex& left, const Complex& right) {  double newX = (left.x \* right.x + left.y \* right.y)/(right.x \* right.x + right.y \* right.y);  double newY = (-(left.x \* right.y) + right.x \* left.y) / (right.x \* right.x + right.y \* right.y);  Complex result(newX, newY);  return result;  }  ostream& operator<<(ostream& left, const Complex& right) {  char l = NULL, r = NULL;  if (right.x >= 0)  l = '+';  if (right.y >= 0)  r = '+';  left << "(" << l<<fixed << setprecision(2) << right.x << ") + i("<<r << fixed << setprecision(2) << right.y << ")" << endl;  return left;  }  Complex& Complex :: operator+=(const Complex right) {  x = x + right.x;  y = y + right.y;  return \*this;  }  Complex& Complex :: operator-=(const Complex right)  {  x = x - right.x;  y = y - right.y;  return \*this;  }  Complex& Complex :: operator\*=(const Complex right)  {  double myx = x; double myy = y;  x = myx \* right.x - myy \* right.y;  y = myx \* right.y + right.x \* myy;  return \*this;  }  Complex& Complex :: operator/=(const Complex right)  {  double myx = x; double myy = y;  x = (myx \* right.x + myy \* right.y)/ (right.x \* right.x + right.y \* right.y);  y = (-(myx \* right.y) + right.x \* myy) / (right.x \* right.x + right.y \* right.y);  return \*this;  }  int main(void) {  Complex c1 = Complex(2.00, 3.00);  Complex c2 = Complex(2.00, 3.00);  Complex c3 = Complex(2.00, 3.00);  Complex c4 = Complex(2.00, 3.00);  Complex r = Complex(1.00, 2.00);  cout << "complex1: " << c1;  cout << "complex2: " << c2;  cout << "complex3: " << c3;  cout << "complex4: " << c4;  cout << "rvalue: " << r;  cout << "complex1 += rvalue: " << (c1 += r);  cout << "complex2 -= rvalue: " << (c2 -= r);  cout << "complex3 \*= rvalue: " << (c3 \*= r);  cout << "complex4 /= rvalue: " << (c4 /= r);  cout << "resulting complex1 + rvalue: " << (c1 + r);  cout << "resulting complex2 - rvalue: " << (c2 - r);  cout << "resulting complex3 \* rvalue: " << (c3 \* r);  cout << "resulting complex4 / rvalue: " << (c4 / r);  cout << endl;  Complex c5 = Complex(19.86, 6.22);  Complex c6 = Complex(19.86, 6.22);  Complex c7 = Complex(19.86, 6.22);  Complex c8 = Complex(19.86, 6.22);  Complex r2 = Complex(0.01, -3.80);  cout << "complex5: " << c5;  cout << "complex6: " << c6;  cout << "complex7: " << c7;  cout << "complex8: " << c8;  cout << "rvalue: " << r2;  cout << "complex5 += rvalue: " << (c5 += r2);  cout << "complex6 -= rvalue: " << (c6 -= r2);  cout << "complex7 \*= rvalue: " << (c7 \*= r2);  cout << "complex8 /= rvalue: " << (c8 /= r2);  cout << "resulting complex5 + rvalue: " << (c5 + r2);  cout << "resulting complex6 - rvalue: " << (c6 - r2);  cout << "resulting complex7 \* rvalue: " << (c7 \* r2);  cout << "resulting complex8 / rvalue: " << (c8 / r2);  cout << endl;  Complex c9 = Complex(1.01, 0.00);  Complex c10 = Complex(1.01, 0.00);  Complex c11 = Complex(1.01, 0.00);  Complex c12 = Complex(1.01, 0.00);  Complex r3 = Complex(-1.01, 1.01);  cout << "complex9: " << c9;  cout << "complex10: " << c10;  cout << "complex11: " << c11;  cout << "complex12: " << c12;  cout << "rvalue: " << r3;  cout << "complex9 += rvalue: " << (c9 += r3);  cout << "complex10 -= rvalue: " << (c10 -= r3);  cout << "complex11 \*= rvalue: " << (c11 \*= r3);  cout << "complex12 /= rvalue: " << (c12 /= r3);  cout << "resulting complex9 + rvalue: " << (c9 + r3);  cout << "resulting complex10 - rvalue: " << (c10 - r3);  cout << "resulting complex11 \* rvalue: " << (c11 \* r3);  cout << "resulting complex12 / rvalue: " << (c12 / r3);  return 0;  } |
| **Output** |
| **Question2** |
| **Source code** |
| #include<iostream>  #include<string>  using namespace std;  class Set {  private:  int\* p;  int size;  public:  Set();  ~Set();  Set(const Set& s);  Set& operator+=(const int right);  Set& operator-=(const int right);  friend const Set operator+(const Set& left, const Set& right);  friend const Set operator\*(const Set& left, const Set& right);  friend const Set operator-(const Set& left, const Set& right);  friend ostream& operator<<(ostream& left, const Set& right);  };  Set::Set():size(0){  p = new int[size];  }  Set::~Set() {  delete p;  }  Set:: Set(const Set& s):p(s.p), size(s.size){}  Set& Set::operator+=(const int right) {  int i = 0;  for (i = 0; i < size; i++) {  if (p[i] == right)  return \*this;  }  int\* newp = new int[size + 1];  for (i = 0; i < size; i++)  newp[i] = p[i];  newp[i] = right;  p = newp;  size++;  return \*this;  }  Set& Set::operator-=(const int right) {  int i = 0; bool b = false; int index = size + 1;  for (i = 0; i < size; i++) {  if (p[i] == right) {  index = i;  b = true;  break;  }  }  if (!b)  return \*this;  int\* newp = new int[size - 1];  int count = 0;  for (i = 0; i < size; i++) {  if (i != index)  newp[count++] = p[i];  }  p = newp;  size--;  return \*this;  }  ostream& operator<<(ostream& left, const Set& right) {  string str = "";  for (int i = 0; i < right.size; i++)  str += to\_string(right.p[i]) + " ";  left << str;  return left;  }  const Set operator+(const Set& left, const Set& right) {  Set newset;  int count = 0; bool b;  int\* newp = new int[left.size + right.size];  for (int j = 0; j < left.size; j++)  newp[count++] = left.p[j];  for (int i = 0; i < right.size; i++) {  b = false;  for (int j = 0; j < left.size; j++) {  if (right.p[i] == left.p[j]) {  b = true;  break;  }  }  if (!b)  newp[count++] = right.p[i];  }  newset.p = newp;  newset.size = count;  return newset;  }  const Set operator\*(const Set& left, const Set& right) {  Set newset;  int\* arr = new int[right.size];  int count = 0; bool b;  for (int i = 0; i < right.size; i++) {  b = false;  for (int j = 0; j < left.size; j++) {  if (left.p[j] == right.p[i]) {  b = true; break;  }  }  if (b) {  arr[count++] = right.p[i];  }  }  for (int i = 0; i < count; i++)  newset += arr[i];  newset.size = count;  return newset;  }  const Set operator-(const Set& left, const Set& right) {  Set newset;  int\* arr = new int[left.size];  int count = 0; bool b;  for (int i = 0; i < left.size; i++) {  b = false;  for (int j = 0; j < right.size; j++) {  if (right.p[j] == left.p[i]) {  b = true; break;  }  }  if (!b) {  arr[count++] = left.p[i];  }  }  for (int i = 0; i < count; i++)  newset += arr[i];  newset.size = count;  return newset;  }  int main(void) {  Set set1;  cout << "Set1: " << set1 << endl;  set1 += 19; set1 += 10; set1 += 17;  cout << "Set1: " << set1 << endl;  set1 += 20; set1 += 21; set1 += 13; set1 += 14;  cout << "Set1: " << set1 << endl;  set1 -= 20; set1 -= 21;  cout << "Set1: " << set1 << endl;  Set set2;  cout << "Set2: " << set2 << endl;  set2 += 22; set2 += 11; set2 += 20;  cout << "Set2: " << set2 << endl;  set2 += 16; set2 += 13; set2 += 14; set2 += 23;  cout << "Set2: " << set2 << endl;  set2 -= 22; set2 -= 23;  cout << "Set2: " << set2 << endl;  cout << "Union of Set1 and Set2: " << (set1 + set2) << endl;  cout << "Difference of Set1 and Set2: " << (set1 - set2) << endl;  cout << "Difference of Set2 and Set1: " << (set2 - set1) << endl;  cout << "Intersection of Set1 and Set2: " << (set1 \* set2) << endl << endl;    Set set3;  cout << "Set3: " << set3 << endl;  set3 += 1; set3 += 2; set3 += 3; set3 += 4; set3 += 5;  cout << "Set3: " << set3 << endl;  set3 += 5; set3 += 6; set3 += 7; set3 += 8;  cout << "Set3: " << set3 << endl;  set3 -= 0; set3 -= 2; set3 -= 4; set3 -= 6; set3 -= 8; set3 -= 9;  cout << "Set3: " << set3 << endl;  Set set4;  cout << "Set4: " << set4 << endl;  set4 += 11; set4 += 42; set4 += 56; set4 += 37;  cout << "Set4: " << set4 << endl;  set4 += 89; set4 += 42; set4 += 56; set4 += 23;  cout << "Set4: " << set4 << endl;  set4 -= 89; set4 += 42; set4 -= 56; set4 += 100; set4 -= 22;  cout << "Set4: " << set4 << endl;  cout << "Union of Set3 and Set4: " << (set3 + set4) << endl;  cout << "Difference of Set3 and Set4: " << (set3 - set4) << endl;  cout << "Difference of Set4 and Set3: " << (set4 - set3) << endl;  cout << "Intersection of Set3 and Set4: " << (set3 \* set4) << endl<<endl;  Set set5;  cout << "Set5: " << set5 << endl;  set5 += 1; set5 += -1;  cout << "Set5: " << set5 << endl;  set5 += 1000000; set5 += -1000000;  cout << "Set5: " << set5 << endl;  set5 -= 1000000; set5 -= -1000000; set5 += 0;  cout << "Set5: " << set5 << endl;  set5 -= 0;  cout << "Set5: " << set5 << endl;  Set set6;  cout << "Set6: " << set6 << endl;  set6 += 164; set6 += 0; set6 += 1986; set6 += 6; set6 += 22;  cout << "Set6: " << set6 << endl;  set6 -= 164; set6 -= 2000;  cout << "Set6: " << set6 << endl;  set6 += 13; set6 += 1;  cout << "Set6: " << set6 << endl;  cout << "Union of Set5 and Set6: " << (set5 + set6) << endl;  cout << "Difference of Set5 and Set6: " << (set5 - set6) << endl;  cout << "Difference of Set6 and Set5: " << (set6 - set5) << endl;  cout << "Intersection of Set5 and Set6: " << (set5 \* set6) << endl;  return 0;  } |
| **Output** |