**Homework 6 컴퓨터공학부 202211390 최준원**

|  |
| --- |
| Q1 |
| Source Code |
| #include <iostream>  #include <iomanip>  #include <string>  using namespace std;    class Complex {  private:  double x;  double y;  public:  Complex();  Complex(double first, double second);  Complex(const Complex& com);  ~Complex();  //Declaration of += operator  Complex& operator+=(const Complex& right);  //Declaration of -= operator  Complex& operator-=(const Complex& right);  //Declaration of \*= operator  Complex& operator\*=(const Complex& right);  //Declaration of /= operator  Complex& operator/=(const Complex& right);  //Declaration of + operator  friend const Complex operator+(const Complex& left, const Complex& right);  //Declaration of - operator  friend const Complex operator-(const Complex& left, const Complex& right);  //Declaration of \* operator  friend const Complex operator\*(const Complex& left, const Complex& right);  //Declaration of / operator  friend const Complex operator/(const Complex& left, const Complex& right);  //Declaration of >> operator  friend istream& operator >> (istream& left, Complex& right);  //Definition of << operator  friend ostream& operator << (ostream& left, const Complex& right);  };  int main() {  Complex complex1(2.00, 3.00);  Complex complex2(2.00, 3.00);  Complex complex3(2.00, 3.00);  Complex complex4(2.00, 3.00);  Complex rvalue(1.00, 2.00);  cout << fixed;  cout << setprecision(2);  cout << "complex1: " << complex1;  cout << "complex2: " << complex2;  cout << "complex3: " << complex3;  cout << "complex4: " << complex4;  cout << "rvalue: " << rvalue;  cout << "complex1 += rvalue: " << (complex1 += rvalue);  cout << "complex2 -= rvalue: " << (complex2 -= rvalue);  cout << "complex3 \*= rvalue: " << (complex3 \*= rvalue);  cout << "complex4 /= rvalue: " << (complex4 /= rvalue) ;  cout << "resulting complex1 + rvalue: " << complex1 + rvalue;  cout << "resulting complex2 - rvalue: " << complex2 - rvalue;  cout << "resulting complex3 \* rvalue: " << complex3 \* rvalue;  cout << "resulting complex4 / rvalue: " << complex4 / rvalue;  cout << endl << "#-- Custom Test Cases " << endl << endl;  Complex complex5(213.00, 322.00);  Complex complex6(32.23, 37.00); //실수부 실수  Complex complex7(-1232.00, 23.00); //실수부 작은 음수  Complex complex8(72.00, 47.42); //허수부 실수  Complex rvalue2(12.00, -2.00);  cout << "complex5: " << complex5;  cout << "complex6: " << complex6;  cout << "complex7: " << complex7;  cout << "complex8: " << complex8;  cout << "rvalue2: " << rvalue2;  cout << "complex5 += rvalue2: " << (complex5 += rvalue2);  cout << "complex6 -= rvalue2: " << (complex6 -= rvalue2);  cout << "complex7 \*= rvalue2: " << (complex7 \*= rvalue2);  cout << "complex8 /= rvalue2: " << (complex8 /= rvalue2);  cout << "resulting complex5 + rvalue2: " << complex5 + rvalue2;  cout << "resulting complex6 - rvalue2: " << complex6 - rvalue2;  cout << "resulting complex7 \* rvalue2: " << complex7 \* rvalue2;  cout << "resulting complex8 / rvalue2: " << complex8 / rvalue2;  cout << endl;    Complex complex9(0.00, 3.00); //실수부 0  Complex complex10(24.00, -12213.00);//허수부 작은 음수  Complex complex11(8.00, 0.00); //허수부 0  Complex complex12(-46432.00, -1243.00); //실허수부 작은 음수  Complex rvalue3(1.23, 73.04); //rvalue 실허수부 실수  cout << "complex9: " << complex9;  cout << "complex10: " << complex10;  cout << "complex11: " << complex11;  cout << "complex12: " << complex12;  cout << "rvalue3: " << rvalue3;  cout << "complex9 += rvalue3: " << (complex9 += rvalue3);  cout << "complex10 -= rvalue3: " << (complex10 -= rvalue3);  cout << "complex11 \*= rvalue3: " << (complex11 \*= rvalue3);  cout << "complex12 /= rvalue3: " << (complex12 /= rvalue3);  cout << "resulting complex9 + rvalue3: " << complex9 + rvalue3;  cout << "resulting complex10 - rvalue3: " << complex10 - rvalue3;  cout << "resulting complex11 \* rvalue3: " << complex11 \* rvalue3;  cout << "resulting complex12 / rvalue3: " << complex12 / rvalue3;  cout << endl;  }  //Complex Class Definition  Complex::Complex()  : x(0), y(0){  }  Complex::Complex(double first, double second)  : x(first), y(second) {  }  Complex::Complex(const Complex& com)  : x(com.x), y(com.y) {  }  Complex::~Complex() {  }  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 first, second;  first = x \* right.x - y \* right.y;  second = x \* right.y + right.x \* y;  x = first;  y = second;  return \*this;  }  Complex& Complex :: operator/=(const Complex& right)  {  double first, second, denom = right.x \* right.x + right.y \* right.y;  first = (x \* right.x + y \* right.y) / denom;  second = (-x \* right.y + right.x \* y) / denom;  x = first;  y = second;  return \*this;  }  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;  }  istream& operator>>(istream& left, Complex& right)  {  cout << "Enter the value of x: ";  left >> right.x;  cout << "Enter the value of y: ";  left >> right.y;  return left;  }  ostream& operator<<(ostream& left, const Complex& right)  {  //left << "(" << right.x << ") + i(" << right.y << ")" << endl;  left << "(";  if (right.x > 0) {  left << "+";  }  left << right.x << ") + i(";  if (right.y > 0) {  left << "+";  }  left << right.y << ")" << endl;  return left;  } |
| Screenshot |
| Question Test Case/  텍스트, 스크린샷, 폰트, 흑백이(가) 표시된 사진  자동 생성된 설명  My Test Case/  텍스트, 스크린샷, 메뉴, 폰트이(가) 표시된 사진  자동 생성된 설명 |

|  |
| --- |
| Q2 |
| Code |
| #include <iostream>  #include <iomanip>  #include <string>  using namespace std;  class Set {  private:  int size;  int\* arr;  public:  Set(int s);  Set();  Set(const Set& set);  ~Set();  Set& operator+=(int element);  Set& operator-=(int element);  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);  };  int main() {  Set set1(0);  Set set2(0);  cout << "set1: " << set1;  set1 += 19;  set1 += 10;  set1 += 17;  cout << "set1: " << set1;  set1 += 20;  set1 += 21;  set1 += 13;  set1 += 14;  cout << "set1: " << set1;  set1 -= 20;  set1 -= 21;  cout << "set1: " << set1;  cout << "set2: " << set2;  set2 += 22;  set2 += 11;  set2 += 20;  cout << "set2: " << set2;  set2 += 16;  set2 += 13;  set2 += 14;  set2 += 23;  cout << "set2: " << set2;  set2 -= 22;  set2 -= 23;  cout << "set2: " << set2;  cout << "Union of set1 and set2: " << set1 + set2;  cout << "Difference of set1 and set2: " << set1 - set2;  cout << "Difference of set2 and set1: " << set2 - set1;  cout << "Intersection of set1 and set2: " << set1 \* set2;    cout << endl << "#-- Custom Test Cases --" << endl;  cout << "#1. 다른 자릿수의 정수와 음수 대입하는 케이스" << endl;  Set set3(0);  Set set4(0);  cout << "set3: " << set3;  set3 += 100; //세자릿수 넣기  set3 += 12;  set3 += 95;  cout << "set3: " << set3;  set3 += 43;  set3 += 23;  set3 += 55;  set3 += 21; //음수 넣기  cout << "set3: " << set3;  set3 -= 100; //세자릿수 빼기  set3 -= -21; //음수 빼기  cout << "set3: " << set3;  cout << "set4: " << set4;  set4 += 12;  set4 += 23;  set4 += -1245;  cout << "set4: " << set4;  set4 + 1234;  set4 += 34;  set4 += 55;  set4 += 43;  cout << "set4: " << set4;  set4 -= 32; //없는 숫자 빼기(실제로는 아무것도 안함)  set4 -= 1234;  cout << "set4: " << set4;  cout << "Union of set3 and set4: " << set3 + set4;  cout << "Difference of set3 and set4: " << set3 - set4;  cout << "Difference of set4 and set3: " << set4 - set3;  cout << "Intersection of set3 and set4: " << set3 \* set4;  cout << endl;  cout << "#2. 빈 Set에 대한 연산 수행" << endl << endl;  cout << " -1. Left Set이 비어있을 경우" << endl << endl;  Set set5(0);  Set set6(0);  cout << "set5: " << set5;  cout << "set6: " << set6;  set6 += 23;  set6 += 23; //같은 거 넣기  set6 += 23; //같은 거 넣기 2트  cout << "set6: " << set6;  set6 += 23;  set6 += 25;  set6 += 24;  set6 -= 23; //빼기  cout << "set6: " << set6;  cout << "Union of set5 and set6: " << set5 + set6;  cout << "Difference of set5 and set6: " << set5 - set6;  cout << "Difference of set6 and set5: " << set6 - set5;  cout << "Intersection of set5 and set6: " << set5 \* set6;  cout << endl;  cout << " -2. Right Set이 비어있을 경우" << endl << endl;  Set set7(0);  Set set8(0);  cout << "set7: " << set7;  set7 += 12;  set7 += 13;  set7 += 14;  cout << "set7: " << set7;  set7 += 15;  set7 += 16;  set7 += 17;  set7 -= 12;  cout << "set7: " << set7;  cout << "set8: " << set8;  cout << "Union of set7 and set8: " << set7 + set8;  cout << "Difference of set7 and set8: " << set7 - set8;  cout << "Difference of set8 and set7: " << set8 - set7;  cout << "Intersection of set7 and set8: " << set7 \* set8;  }  //Set Class Definition  Set::Set(int s)  :size(s)  {  arr = (int\*)malloc(sizeof(int) \* size);  }  Set::Set()  :size(0)  {  arr = (int\*)malloc(sizeof(int) \* size);  }  Set::Set(const Set& set)  {  size = set.size;  arr = (int\*)realloc(arr, size \* sizeof(int));  for (int i = 0; i < size; i++) {  arr[i] = set.arr[i];  }  }  Set::~Set()  {  free(arr);  }  Set& Set::operator+=(int element)  {  int\* tempArr = 0;  tempArr = (int\*)malloc(sizeof(int) \* size);  for (int i = 0; i < size; i++) {  if (arr[i] == element) {  return \*this; //사전에 같은 거 있는지 검사  }  tempArr[i] = arr[i];  }  size++;  arr = (int\*)realloc(arr, size \* sizeof(int));  for (int i = 0; i < size - 1; i++) {  arr[i] = tempArr[i];  //cout << "inserted: " << tempArr[i] << endl;  }  arr[size - 1] = element;  //cout << "inserted: " << element << endl;  free(tempArr);  return \*this;  }  Set& Set::operator-=(int element)  {  bool check = false;  int i, j;  int\* tempArr = 0;  tempArr = (int\*)malloc(sizeof(int) \* size);  for (i = 0; i < size; i++) {  tempArr[i] = arr[i];  }  for (i = 0; i < size; i++) {  if (tempArr[i] == element) {  check = true; //원소 같은 게 아예 없으면 그대로 돌아감  }  }  if (!check) {  free(tempArr);  return \*this;  }  size--;  arr = (int\*)realloc(arr, size \* sizeof(int));  i = 0;  j = 0;  while (1) {  if (tempArr[j] == element) { // 같은 거 있으면 포함 안해요  j++;  continue;  }  //cout << "reloaded: " << tempArr[j] << " " << j << endl;  arr[i] = tempArr[j]; // 중복 아니었던 l집어넣기    i++;  j++;  if (i == size) {  break;  }  }  free(tempArr);  return \*this;  }  const Set operator+(const Set& left, const Set& right)  {  Set returnSet(left.size);  for (int i = 0; i < left.size; i++) {  returnSet.arr[i] = left.arr[i];  }  for (int i = 0; i < right.size; i++) {  returnSet += right.arr[i];  }  return returnSet;  }  const Set operator\*(const Set& left, const Set& right)  {  if (left.size == 0 || right.size ==0) { //둘 중 하나가 비어있으면 공집합 부재  Set nullSet(0);  return nullSet;  }  Set returnSet(left.size);  Set tempSet(left.size);  int\* temp;  int count = 0;  temp = (int\*)malloc(sizeof(int) \* left.size);  for (int i = 0; i < left.size; i++) {  returnSet.arr[i] = left.arr[i];  tempSet.arr[i] = left.arr[i];  }  for (int i = 0; i < right.size; i++) {  tempSet -= right.arr[i]; // left 쪽 비중복  }  for (int i = 0; i < left.size; i++) {  returnSet -= tempSet.arr[i]; // left 쪽 중복  }  return returnSet;  }  const Set operator-(const Set& left, const Set& right)  {  if (left.size == 0) { //왼쪽이 비어있으면 공집합  Set nullSet(0);  return nullSet;  }  if (right.size == 0) { //오른쪽이 비어있으면 왼쪽 반환  Set staticSet(left.size);  for (int i = 0; i < left.size; i++) {  staticSet.arr[i] = left.arr[i];  }  return staticSet;  }  Set returnSet(left.size);  for (int i = 0; i < left.size; i++) {  returnSet.arr[i] = left.arr[i];  }  for (int i = 0; i < right.size; i++) {  returnSet -= right.arr[i];  //cout << "roajtwlsk " << right.arr[i] << endl;  }  return returnSet;  }  ostream& operator<<(ostream& left, const Set& right)  {  for (int i = 0; i < right.size; i++) {  left << right.arr[i] << " ";  }  left << endl;  return left;  } |
| Screenshot |
| Question Test Case/  텍스트, 스크린샷, 폰트, 블랙이(가) 표시된 사진  자동 생성된 설명  My Test Case/  텍스트, 스크린샷, 폰트, 디자인이(가) 표시된 사진  자동 생성된 설명 |