7.7 A set class

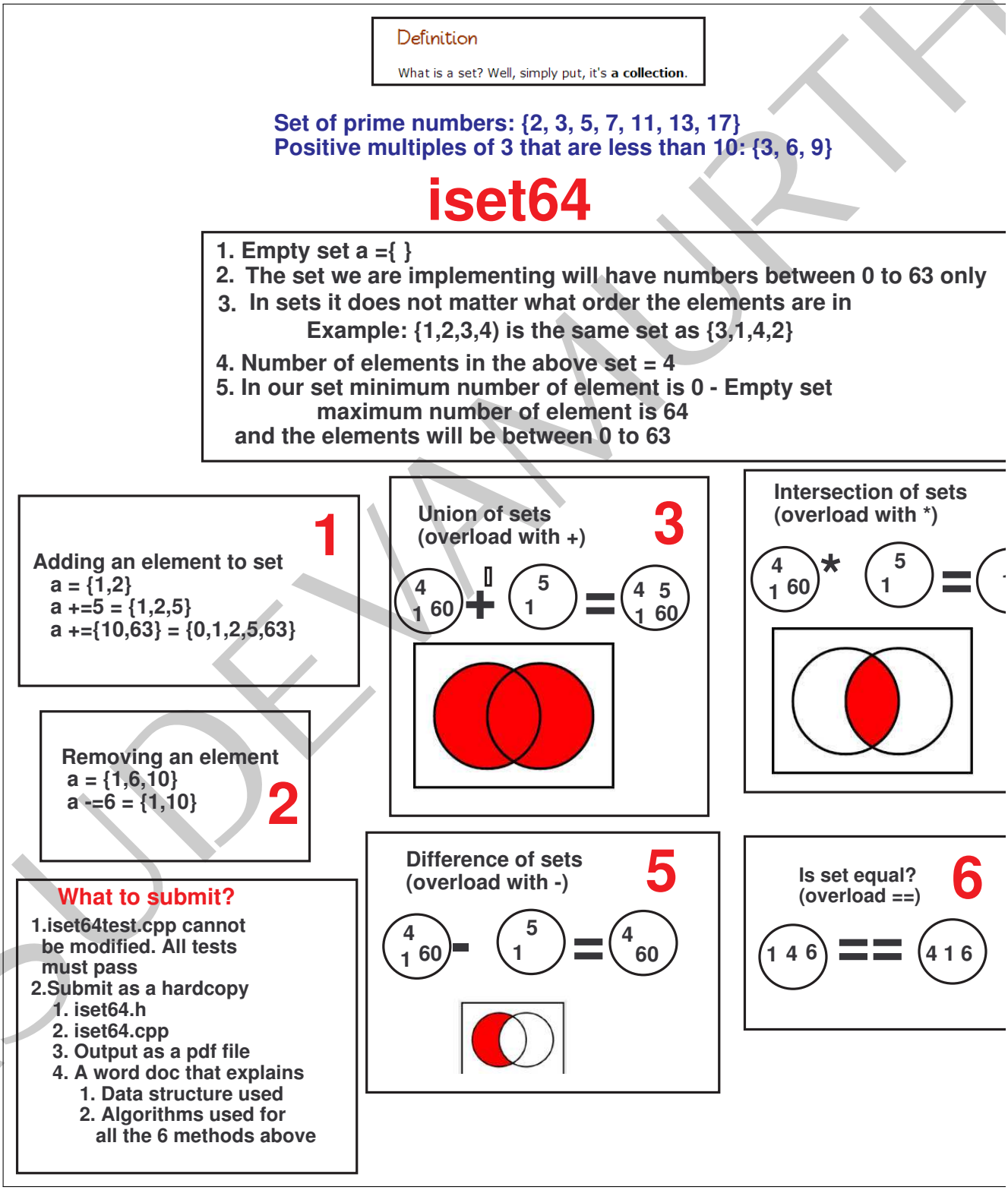

Figure 7.14: Set

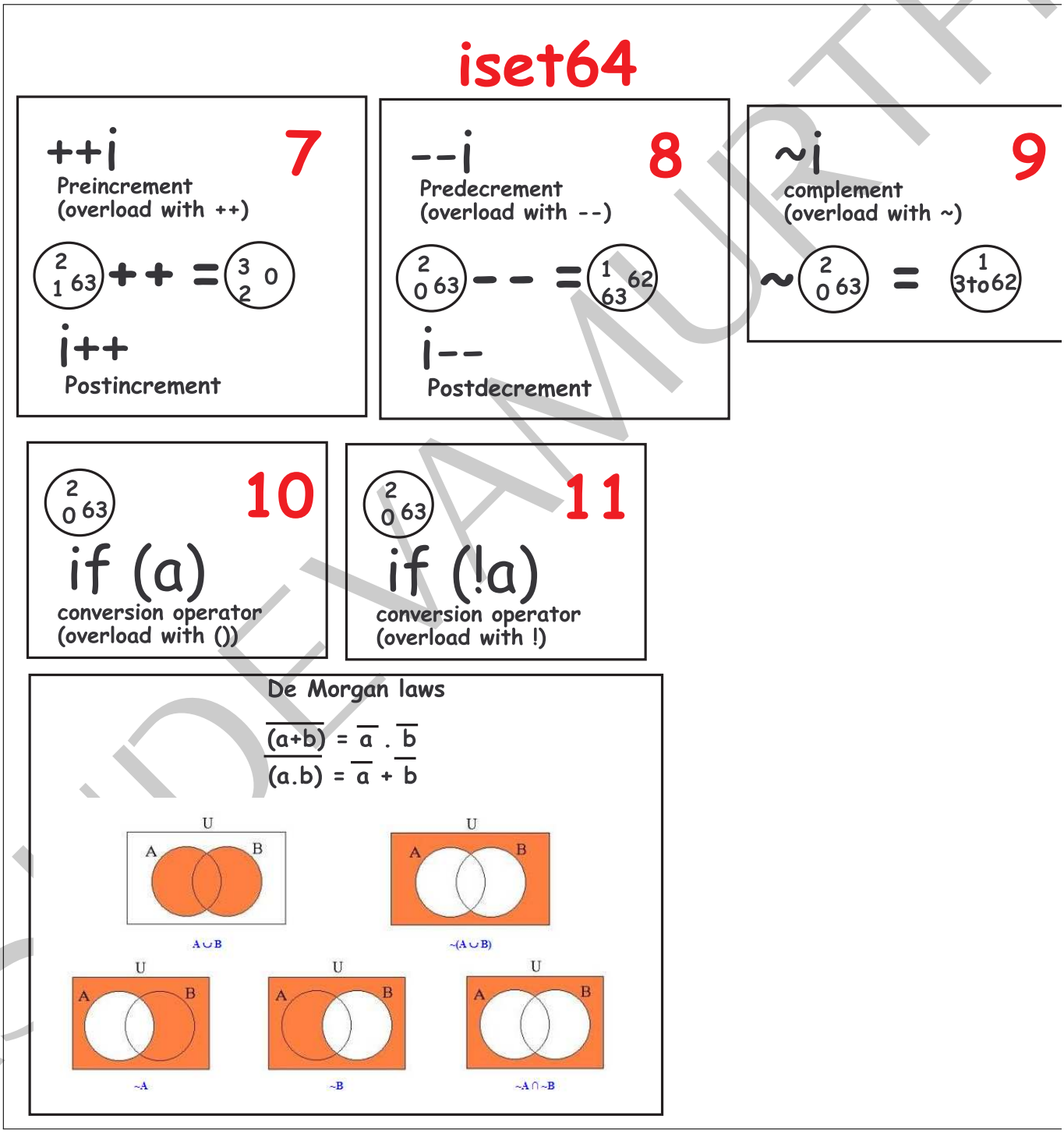

Figure 7.15: Set

```
/*----------------------------------------------------------------
Copyright (c) 2013 Author: Jagadeesh Vasudevamurthy
file: iset64test.cpp

On linux:
g++ iset64.cpp iset64test.cpp
valgrind a.out

----------------------------------------------------------------*/

/*----------------------------------------------------------------
This file test iset64 object
----------------------------------------------------------------*/

/*----------------------------------------------------------------
All includes here
----------------------------------------------------------------*/
#include "iset64.h"

/*----------------------------------------------------------------
test a set
----------------------------------------------------------------*/
void test_basic() {
  iset64 a;
  cout << "a = " << a << endl;
  a = a + 5;
  cout << "set a after adding 5 = " << a << endl;
  a = a + 5;
  cout << "set a after adding 5 = " << a << endl;
  a += 63;
  a += 0;
  cout << "set a after adding 0 and 63 = " << a << endl;
  int x[] = { 1, 3, 6 };
  iset64 b(x, sizeof(x) / sizeof(int));
  cout << "set b = " << b << endl;
  b = b - 3;
  cout << "set b after removing 3 = " << b << endl;
  b = b - 3;
  cout << "set b after removing 3 = " << b << endl;
  b = b - 10;
  cout << "set b after removing 10 = " << b << endl;
  b = b - 6;
  cout << "set b after removing 6 = " << b << endl;
  b = b - 1;
  cout << "set b after removing 1 = " << b << endl;
  b = b + 10;
  b = b + 2;
  cout << "set b after adding {10,2} = " << b << endl;
}

/*----------------------------------------------------------------
test union
----------------------------------------------------------------*/
void test_union() {
  {
    cout << "TESTING: iset64 operator+(const iset64& a, const iset64& b)" << endl;
    iset64 a;
    a += 1;
    a += 2;
    iset64 b;
    b += 1;
    b += 2;
    b += 3;
    cout << "Set a " << a << endl;
    cout << "Set b " << b << endl;
    iset64 c = a + b;
```

```
    cout << "a + b = " << c << endl;
  }
  {
    cout << "TESTING:iset64 operator+(const iset64& a, const int b)" << endl;
    iset64 a;
    a += 1;
    a += 2;
    cout << a << endl;
    a = a + 1;
    cout << "{1,2} + 1 = " << a << endl;
    a += 1;
    a += 2;
    cout << a << endl;
    a = a + 3;
    cout << "{1,2} + 3 = " << a << endl;
  }
  {
    cout << "TESTING:iset64 operator+(const int b, const iset64& a)" << endl;
    iset64 a;
    a += 1;
    a += 2;
    cout << "Set a " << a << endl;
    a = 1 + a;
    cout << " 1 + {1,2} = " << a << endl;
    a += 1;
    a += 2;
    cout << "Set a " << a << endl;
    a = 3 + a;
    cout << " 3 + {1,2}  = " << a << endl;
  }

  {
    cout << "TESTING:iset64& iset64::operator+=(const iset64& a)" << endl;
    iset64  b;
    b += 1;
    b += 2;
    iset64 a;
    a += 1;
    a += 3;
    cout << "Set b " << b << endl;
    cout << "Set a " << a << endl;
    b += a;
    cout << " {1,2} + {1,3}  = " << b << endl;
  }
  {
    cout << "iset64& iset64::operator+=(const int b)" << endl;
    iset64 a;
    a += 1;
    a += 2;
    cout << "Set a " << a << endl;
    a += 3;
    cout << " {1,2} + 3  = " << a << endl;
  }
  {
    //test chaining
    iset64 a;
    a += 1;
    a += 2;
    iset64 b;
    b += 3;
    b += 4;
    iset64 c;
    c += 7;
    c += 8;
    iset64 d = a + b + c + 5;
    cout << "Set a " << a << endl;
```

```
    cout << "Set b " << b << endl;
    cout << "Set c " << c << endl;
    cout << "Set d " << d << endl;
  }
}

/*----------------------------------------------------------------
test difference
----------------------------------------------------------------*/
void test_difference() {
  {
    cout << "TESTING: iset64 operator-(const iset64& a, const iset64& b)" << endl;
    iset64 a;
    a += 1;
    a += 2;
    iset64 b;
    b += 1;
    b += 2;
    iset64 c = a - b;
    cout << "Set a " << a << endl;
    cout << "Set b " << a << endl;
    cout << "a - b = " << c << endl;
  }
  {
    cout << "TESTING: iset64 operator-(const iset64& a, const iset64& b)" << endl;
    iset64 a;
    a += 1;
    a += 5;
    iset64 b;
    b += 1;
    b += 2;
    b += 3;
    iset64 c = a - b;
    cout << "Set a " << a << endl;
    cout << "Set b " << b << endl;
    cout << "a - b = " << c << endl;
  }

  {
    cout << "TESTING: iset64 operator-(const iset64& a, const int b)" << endl;
    iset64 a;
    a += 1;
    a += 2;
    cout << "Set a " << a << endl;
    a = a - 3;
    cout << "a - 3 = " << a << endl;
  }

  {
    cout << "TESTING: iset64 operator-(const int b, const iset64& a)" << endl;
    iset64 a;
    a += 1;
    a += 2;
    cout << "Set a " << a << endl;
    a = 3 - a;
    cout << "3 - a = " << a << endl;
  }

  {
    cout << "TESTING: iset64& iset64::operator-=(const iset64& a)" << endl;
    iset64 a;
    a += 1;
    a += 3;
    iset64 b;
    b += 1;
    b += 2;
```

```
    cout << "Set a " << a << endl;
    cout << "Set b " << b << endl;
    b -= a;
    cout << "b -= a = " << b << endl;
  }

  {
    cout << "TESTING: iset64& iset64::operator-=(const int b)" << endl;
    iset64 a;
    a += 1;
    a += 2;
    cout << "Set a " << a << endl;
    a -= 3;
    cout << "a -= 3 = " << a << endl;
  }
  {
    //test chaining
    iset64 a;
    a += 1;
    a += 2;
    iset64 b;
    b += 2;
    b += 4;
    iset64 c;
    c += 2;
    c += 8;
    iset64 d = a - b - c + 5;
    cout << "Set a " << a << endl;
    cout << "Set b " << b << endl;
    cout << "Set c " << c << endl;
    cout << "Set d " << d << endl;
  }
}

/*-----------------------------------------------------------------
test intersection
-----------------------------------------------------------------*/
void test_intersection() {
  {
    cout << "TESTING: iset64 operator*(const iset64& a, const iset64& b)" << endl;
    iset64 a;
    a += 1;
    a += 2;
    iset64 b;
    b += 1;
    b += 2;
    b += 3;
    cout << "Set a " << a << endl;
    cout << "Set b " << b << endl;
    iset64 c = a * b;
    cout << "a * b = " << c << endl;
  }
  {
    cout << "TESTING:iset64 operator*(const iset64& a, const int b)" << endl;
    iset64 a;
    a += 1;
    a += 2;
    cout << "Set a " << a << endl;
    a = a * 1;
    cout << "{1,2} * 1 = " << a << endl;
    a += 1;
    a += 2;
    cout << "Set a " << a << endl;
    a = a * 3;
    cout << "{1,2} * 3 = " << a << endl;
  }
```

```
  {
    cout << "TESTING:iset64 operator*(const int b, const iset64& a)" << endl;
    iset64 a;
    a += 1;
    a += 2;
    cout << "Set a " << a << endl;
    a = 1 * a;
    cout << " 1 * {1,2} = " << a << endl;
    a += 1;
    a += 2;
    cout << "Set a " << a << endl;
    a = 3 * a;
    cout << " 3 * {1,2}  = " << a << endl;
  }

  {
    cout << "TESTING:iset64& iset64::operator*=(const iset64& a)" << endl;
    iset64 b;
    b += 1;
    b += 2;
    iset64 a;
    a += 1;
    a += 3;
    cout << "Set b " << b << endl;
    cout << "Set a " << a << endl;
    b *= a;
    cout << " {1,2} * {1,3}  = " << b << endl;
  }
  {
    cout << "iset64& iset64::operator*=(const int b)" << endl;
    iset64 a;
    a += 1;
    a += 2;
    cout << "Set a " << a << endl;
    a *= 3;
    cout << " {1,2} * 3  = " << a << endl;
  }
  {
    //test chaining
    iset64 a;
    a += 1;
    a += 2;
    iset64 b;
    b += 2;
    b += 4;
    iset64 c;
    c += 2;
    c += 8;
    iset64 d = a * b * c + 5;
    cout << "Set a " << a << endl;
    cout << "Set b " << b << endl;
    cout << "Set c " << c << endl;
    cout << "Set d " << d << endl;
  }
}


/*-----------------------------------------------------------------
test equal
-----------------------------------------------------------------*/
void test_equal_not_equal() {
  {
    cout << "TESTING: bool operator==(const iset64& a, const iset64& b)" << endl;
    iset64 a;
    a += 1;
    a += 2;
```

```
    iset64 b;
    b += 1;
    b += 2;
    cout << "Set a " << a << endl;
    cout << "Set b " << b << endl;
    cout << "a == b " << boolalpha << (a == b) << endl;
    b -= 1;
    cout << a;
    cout << b;
    cout << "a == b " << boolalpha << (a == b) << endl;
  }
  {
    cout << "TESTING: bool operator!=(const iset64& a, const iset64& b)" << endl;
    iset64 a;
    a += 1;
    a += 2;
    iset64 b;
    b += 1;
    b += 2;
    cout << "Set a " << a << endl;
    cout << "Set b " << b << endl;
    cout << "a != b " << boolalpha << (a != b) << endl;
    b -= 1;
    cout << "Set a " << a << endl;
    cout << "Set b " << b << endl;
    cout << "a != b " << boolalpha << (a != b) << endl;
  }
}

/*-----------------------------------------------------------------
++ and --
-----------------------------------------------------------------*/
void test_pre_post_inr_dec() {
  {
    int x[] = { 1, 2, 63 };
    iset64 a(x, sizeof(x) / sizeof(int));
    cout << "a = " << a << endl;
    ++a;
    cout << "++a = " << a << endl;
    int y[] = { 2, 3, 0 };
    iset64 b(y, sizeof(y) / sizeof(int));
    assert(a == b);
  }
  {
    int x[] = { 1, 2, 63 };
    iset64 a(x, sizeof(x) / sizeof(int));
    cout << "a = " << a << endl;
    iset64 acopy(x, sizeof(x) / sizeof(int));
    cout << "acopy = " << acopy << endl;
    iset64 rhs = a++;
    assert(rhs == acopy);
    cout << "a++ = " << a << endl;
    cout << "rhs = " << rhs << endl;
    int y[] = { 2, 3, 0 };
    iset64 b(y, sizeof(y) / sizeof(int));
    assert(a == b);
  }
  {
    int x[] = { 0,2,63 };
    iset64 a(x, sizeof(x) / sizeof(int));
    cout << "a = " << a << endl;
    --a;
    cout << "--a = " << a << endl;
    int y[] = { 63, 1, 62 };
    iset64 b(y, sizeof(y) / sizeof(int));
    assert(a == b);
```

```
  }
  {
    int x[] = { 0, 2, 63 };
    iset64 a(x, sizeof(x) / sizeof(int));
    cout << "a = " << a << endl;
    iset64 acopy(x, sizeof(x) / sizeof(int));
    cout << "acopy = " << acopy << endl;
    iset64 rhs = a--;
    assert(rhs == acopy);
    cout << "a-- = " << a << endl;
    cout << "rhs = " << rhs << endl;
    int y[] = { 63, 1, 62 };
    iset64 b(y, sizeof(y) / sizeof(int));
    assert(a == b);
  }
}

/*-----------------------------------------------------------------
~
Complement of a set.
The complement of A is the set of all element in the universal set U, but not in A.
a = {0,2,63}
x = ~a
{1,3,...,62}
-----------------------------------------------------------------*/
void test_complement() {
  {
    int x[] = { 0, 2, 63 };
    iset64 a(x, sizeof(x) / sizeof(int));
    cout << "a = " << a << endl;
    iset64 nota = (~a);
    cout << "~a = " << nota << endl;
    iset64 ans;
    ans += 1;
    for (int i = 3; i < 63; ++i) {
      ans += i;
    }
    cout << "ans = " << ans << endl;
    assert(nota == ans);
    ans = ~ans;
    cout << "~ans = " << ans << endl;
    assert(ans == a);
  }
}

/*-----------------------------------------------------------------
a = {0,2,63}
if (a) {

}
-----------------------------------------------------------------*/
void test_conversion_operator() {
  int x[] = { 0, 2, 63 };
  iset64 a(x, sizeof(x) / sizeof(int));
  cout << "a = " << a << endl;
  if (a) {
    cout << "a exists\n";
  } else {
    cout << "a does not exists\n";
  }
  iset64 b;
  cout << "b = " << b << endl;
  if (b) {
    cout << "b exists\n";
  } else {
    cout << "b does not exists\n";
```

```
  }
}

/*-----------------------------------------------------------------
a = {0,2,63}
if (!a) {

}
-----------------------------------------------------------------*/
void test_not_operator() {
  int x[] = { 0, 2, 63 };
  iset64 a(x, sizeof(x) / sizeof(int));
  cout << "a = " << a << endl;
  if (!a) {
    cout << "a does not exists\n";
  } else {
    cout << "a exists\n";
  }
  iset64 b;
  cout << "b = " << b << endl;
  if (!b) {
    cout << "b does not exists\n";
  } else {
    cout << "b exists\n";
  }
}

/*-----------------------------------------------------------------
(a+b)' = a'. b'
(a.b)' = a' + b'
-----------------------------------------------------------------*/
void test_demorgan_laws(const int x[], int lx, const int y[], int ly) {
  {
    iset64 a(x, lx);
    cout << "a = " << a << endl;

    iset64 b(y, ly);
    cout << "b = " << b << endl;

    iset64 aplusb = a + b;
    cout << "aplusb = " << aplusb << endl;

    iset64 aplusbbar = ~(aplusb);
    cout << "aplusbbar = " << aplusbbar << endl;

    iset64 abar = ~(a);
    cout << "abar = " << abar << endl;

    iset64 bbar = ~(b);
    cout << "bbar = " << bbar << endl;

    iset64 abarplusbbar = abar + bbar;
    cout << "abarplusbbar = " << abarplusbbar << endl;

    iset64 abardotbbar = abar * bbar;
    cout << "abardotbbar = " << abardotbbar << endl;

    iset64 adotb = a * b;
    cout << "adotb  = " << adotb << endl;

    iset64 adotbbar = ~(adotb);
    cout << "adotbbar  = " << adotbbar << endl;

    assert(aplusbbar == abardotbbar);
    cout << "Demorgan law (a+b)' = a'. b' is proved\n";
    assert(adotbbar == abarplusbbar);
```

```
    cout << "Demorgan law (a.b)' = a' + b' is proved\n";
  }
}

/*-----------------------------------------------------------------
(a+b)' = a'. b'
(a.b)' = a' + b'
-----------------------------------------------------------------*/
void test_demorgan_laws() {
  {
    int x[] = { 4, 5, 6 };
    int y[] = { 5, 6, 8 };
    test_demorgan_laws(x, (sizeof(x) / sizeof(int)), y, (sizeof(y) / sizeof(int)));
  }
  {
    int x[] = { 1,2,4,5 };
    int y[] = { 2,3,5,6 };
    test_demorgan_laws(x, (sizeof(x) / sizeof(int)), y, (sizeof(y) / sizeof(int)));
  }

}

/*-----------------------------------------------------------------
test bed
-----------------------------------------------------------------*/
void testbed() {
  test_basic();
  test_union();
  test_difference();
  test_intersection();
  test_equal_not_equal();
  test_pre_post_inr_dec();
  test_complement();
  test_conversion_operator();
  test_not_operator();
  test_demorgan_laws();
}

/*-----------------------------------------------------------------
main
-----------------------------------------------------------------*/
int main() {
  testbed();
  return 0;
}

//EOF
```