```
//testing1.cpp
#include <iostream>
using namespace std;
/*Positive + Positive
Zero + Zero
Positive + Zero
Negative + Zero
Positive + Negative
Negative + Positive
Negative + Negative*
Large numbers*/

bool additionTest();

int addition(int a, int b);

int main ( )
{
	if( additionTest()==true) // all the test must pass to be true
		cout  << "passed";
	else
		cout  << "failed";
}

int addition(int a, int b)
{
  int sum = a + b;
  return sum;
}

bool additionTest()
{
	if(addition(1, 1) != 2)
		return (false);
	if(addition(0, 0)!= 0)
		return (false);
	if(addition(2, 0)!= 2)
		return (false);
	if(addition(-2, 0)!= -2)
		return (false);
```

```
        if(addition(1, -2)!= -1)
                return (false);
        if(addition(-2, 1)!= -1)
                return (false);
        if(addition(-2, -1)!= -3)
                return (false);
        if(addition(1234, 988)!= 2222)
                return (false);
        if(addition(-1234,-988)!= -2222)
                return (false);
        return (true);
```

Output
passed

//testing2.cpp

```
#include <iostream>
using namespace std;

/*Positive + Positive
Zero + Zero
Positive + Zero
Negative + Zero
Positive + Negative
Negative + Positive
Negative + Negative*
Large numbers*/

bool additionTest();

int addition(int a, int b);

int main ( )
{
        if( additionTest()==true) // all the test must pass to be true
                cout  << "passed";
        else
                cout  << "failed";
}
```

```
int addition(int a, int b)
{
   int sum = a + b;
   return sum;
}

bool additionTest()
{
        if(addition(1, 1) != 2)
                return (false);
        if(addition(0, 0)!= 0)
                return (false);
        if(addition(2, 0)!= 2)
                return (false);
        if(addition(-2, 0)!= -2)
                return (false);
        if(addition(1, -2)!= -1)
                return (false);
        if(addition(-2, 1)!= -1)
                return (false);
        if(addition(-2, -1)!= -3)
                return (false);
        if(addition(1234, 988)!= 2222)
                return (false);
        if(addition(-1234,-988)!= -333333) // failed
                return (false);
        return (true);

}
```

Output

failed

}

// testing3

```
#include <iostream>
using namespace std;

/*Positive + Positive
Zero + Zero
Positive + Zero
Negative + Zero
Positive + Negative
Negative + Positive
Negative + Negative*
Large numbers*/

bool additionPropertiesTest(); // prototype

int addition(int a, int b);

int main ( )
{
        if( additionPropertiesTest()==true) // all the test must pass to be true
                cout  << "passed";
        else
                cout  << "failed";
}

int addition(int a, int b)
{
   int sum = a + b;
   return sum;
}

bool additionPropertiesTest()
{
   // commutative: a + b = b + a
   if ( addition(1, 2) != addition(2, 1) )
      return (false);

   // associative: a + (b + c) = (a + b) + c
   if ( addition(1, addition(2, 3)) != addition(addition(1, 2), 3) )
      return (false);
```

```
    // neutral element: a + NEUTRAL = a
    if ( addition(10, 0) != 10 )
        return (false);

    // inverse element: a + INVERSE = NEUTRAL
    if ( addition(10, -10) != 0 )
        return (false);

    return (true);
}
```

Output

Passed

// testing4

```
#include <iostream>
using namespace std;

/*Positive + Positive
Zero + Zero
Positive + Zero
Negative + Zero
Positive + Negative
Negative + Positive
Negative + Negative*
Large numbers*/

bool additionPropertiesTest(); // prototype

int addition(int a, int b); // prototype

int main ( )
{
        if( additionPropertiesTest()==true)
                cout  << "passed";
        else
                cout  << "failed";
}

int addition(int a, int b)
```

```
{
    int sum = a + b;
    return sum;
}

bool additionPropertiesTest()
{
    // conmutative: a + b = b + a
    if ( addition(1, 2) != addition(2, 1) )
        return (false);

    // asociative: a + (b + c) = (a + b) + c
    if ( addition(1, addition(2, 3)) != addition(addition(1, 2), 3) )
        return (false);

    // neutral element: a + NEUTRAL = a
    if ( addition(10, 0) != 10 )
        return (false);

    // inverse element: a + INVERSE = NEUTRAL
    if ( addition(10, -10) != 5 ) // failed
        return (false);

    return (true);
}
```

Output

Failed

```
//testing5.cpp

class  addClass
{
public :
        int adder ( int  first , int  second ) ;
 private :
        int          firstNumber ;
        int          secondNumber ;
} ;

int  addClass::adder( int  first , int  second )
        {
        firstNumber = first ;
        secondNumber = second ;
        int sum = firstNumber+ secondNumber;
        return sum;
        }

#include <iostream>
#include <cassert>
using namespace std;

/*Positive + Positive
Zero + Zero
Positive + Zero
Negative + Zero
Positive + Negative
Negative + Positive
Negative + Negative*
Large numbers*/

int main ( )
{
        addClass mySum;
        assert(mySum.adder(1, 1) == 2);
        assert(mySum.adder(0, 0) == 0);
        assert(mySum.adder(2, 0) == 2);
        assert(mySum.adder(-2, 0) == -2);
        assert(mySum.adder(1, -2) == -1);
```

```
        assert(mySum.adder(-2, 1) == -1);
        assert(mySum.adder(-2, -1) == -3);
        assert(mySum.adder(1234, 988) == 2222);
        assert(mySum.adder(-1234,-988) == -2222);
        assert(mySum.adder(1, 1) == 222);

}
```

Output

```
Assertion failed: mySum.adder(1, 1) == 222, file ..\testing5.cpp, line 44

This application has requested the Runtime to terminate it in an unusual way.
Please contact the application's support team for more information.
```