```
/* FILE NAME:  …….
 * AUTHOR: Solution Briefing
 * See our syllabus for a good comment
 */

#include <iostream>

#include <stdlib.h>

#include <assert.h>

#include <ctime>

using namespace std;

/*

* Input: A_alive indicates Aaron is alive true for alive, false for dead

*            B_alive indicates Bob is alive

*            C_alive indicates Charlie is alive

* Return: true if at least two are alive otherwise return false

*/

bool at_least_two_alive(bool A_alive, bool B_alive, bool C_alive);


/*

* Call by reference

* Strategy 1: Everyone shoots to kill the highest accuracy player alive

* Input: B_alive indicates Bob is alive or dead

*            C_alive indicates Aaron is alive or dead

* Return: Change B_alive into false if Bob is killed

*             Change C_alive into false if Charlie is killed

*/

void Aaron_shoots1(bool& B_alive, bool& C_alive);


/*

* Call by reference

* Input: A_alive indicates Aaron is alive or dead

*            C_alive indicates Charlie is alive or dead

* Return: Change A_alive into false if Aaron is killed

*             Change C_alive into false if Charlie is killed
```

```
*/
void Bob_shoots(bool& A_alive, bool& C_alive);

/*
* Call by reference
* Input: A_alive indicates Aaron is alive or dead
*            B_alive indicates Bob is alive or dead
* Return: Change A_alive into false if Aaron is killed
*             Change B_alive into false if Bob is killed
*/
void Charlie_shoots(bool& A_alive, bool& B_alive);

/*
* Call by reference
* Strategy 2: Aaron intentionally misses if both are alive
* Input: B_alive indicates Bob is alive or dead
*            C_alive indicates Aaron is alive or dead
* Return: Change B_alive into false if Bob is killed
*             Change C_alive into false if Charlie is killed
*/
void Aaron_shoots2(bool& B_alive, bool& C_alive);

//Simple method to implement pause function in linux
void Press_any_key(void);

//TEST PROTOTYPES
void test_at_least_two_alive(void);
void test_Aaron_shoots1(void);
void test_Bob_shoots(void);
void test_Charlie_shoots(void);
void test_Aaron_shoots2(void);
```

```
//VARIABLES
……

int main() {
	//Initializes Random number generator's seed and calls test functions
	cout << "*** Welcome to Li's Duel Simulator ***\n";
	srand(time(0));
	test_at_least_two_alive();
	Press_any_key();
	test_Aaron_shoots1();
	Press_any_key();
	test_Bob_shoots();
	Press_any_key();
	test_Charlie_shoots();
	Press_any_key();
	test_Aaron_shoots2();
	Press_any_key();

	//Starts strategy 1 and runs 10,000 times
	cout << "Ready to test strategy 1 (run 10000 times):\n";
	Press_any_key();
	for (int i = 0; i < TOTAL_RUNS; i++ ){
		……..
		while (at_least_two_alive(aaronAlive, bobAlive, charlieAlive)) {
			……..
			}
		}
		if (aaronAlive)
			…….
		if (bobAlive)
			……..
		if (charlieAlive)
```

```
                    ……..
        }
        cout << "Aaron won " << aaronWins1 << "/10000 duels or " << static_cast<double>(aaronWins1) /
TOTAL_RUNS * 100 << "%\n"
            << "Bob won " << bobWins << "/10000 duels or " << static_cast<double>(bobWins) / TOTAL_RUNS
* 100 << "%\n"
            << "Charlie won " << charlieWins << "/10000 duels or " << static_cast<double>(charlieWins) /
TOTAL_RUNS * 100 << "%\n"
            << endl;

        //Reinitializes variables and starts strategy 2 to run 10,000 times
        ……..
        cout << "Ready to test strategy 2 (run 10000 times):\n";
        Press_any_key();
        for (int i = 0; i < TOTAL_RUNS; i++ ){
                ……
                while (at_least_two_alive(aaronAlive, bobAlive, charlieAlive)) {
                        …….
                }
                if (aaronAlive)
                        …….
                if (bobAlive)
                        ……..
                if (charlieAlive)
                        ……
        }
        cout << "Aaron won " << aaronWins2 << "/10000 duels or " << static_cast<double>(aaronWins2) /
TOTAL_RUNS * 100 << "%\n"
            << "Bob won " << bobWins << "/10000 duels or " << static_cast<double>(bobWins) / TOTAL_RUNS
* 100 << "%\n"
            << "Charlie won " << charlieWins << "/10000 duels or " << static_cast<double>(charlieWins) /
TOTAL_RUNS * 100 << "%\n"
            << endl;
```

```
	if (…….) {
		cout << "Strategy 1 is better than strategy 2.\n";
	}
	else {
		cout << "Strategy 2 is better than strategy 1.\n";
	}

	return 0;
}

//Implementation of functions. Look above for documentation of them.
bool at_least_two_alive(bool A_alive, bool B_alive, bool C_alive) {
	…….
}

void test_at_least_two_alive(void) {
	cout << "Unit Testing 1: Function - at_least_two_alive()\n";

	cout << "\tCase 1: Aaron alive, Bob alive, Charlie alive\n";
	assert(…….);
	cout << "\tCase passed ...\n";

	cout << "\tCase 2: Aaron dead, Bob alive, Charlie alive\n";
	assert(……);
	…….
	assert(……);
	…….
	assert(……);
	…….

	……..
	 ……..
```

```
}

void Aaron_shoots1(bool& B_alive, bool& C_alive) {
	assert(……..);
	int shootResult = rand() % 100;
	if (C_alive) {
		…….
	}
	else {
		……
	}
}

void test_Aaron_shoots1(void) {
	cout << "Unit Testing 2: Function Aaron_shoots1(Bob_alive, Charlie_alive)\n";

	……
	cout << "\tCase 1: Bob alive, Charlie alive\n"
	    << "\t\tAaron is shooting at Charlie\n";
	Aaron_shoots1(bob_a, charlie_a);

	bob_a = false;
	charlie_a = true;
	cout << "\tCase 2: Bob dead, Charlie alive\n"
	    << "\t\tAaron is shooting at Charlie\n";
	Aaron_shoots1(bob_a, charlie_a);

	bob_a = true;
	charlie_a = false;
	cout << "\tCase 3: Bob alive, Charlie dead\n"
	    << "\t\tAaron is shooting at Bob\n";
	Aaron_shoots1(bob_a, charlie_a);
```

```
}

void Bob_shoots(bool& A_alive, bool& C_alive) {
        ……
}

void test_Bob_shoots(void) {
        …….

}

void Charlie_shoots(bool& A_alive, bool& B_alive) {
        ……..
}

void test_Charlie_shoots(void) {
        …….
        }

void Aaron_shoots2(bool& B_alive, bool& C_alive) {
        …….
}

void test_Aaron_shoots2(void) {
        ……
}

void Press_any_key(void) {
        cout << "Press any key to continue...";
        cin.ignore().get();
}
```