

Name: Dylan McDonald

Section: *n*

Date: 02 October, 2007

Questions 1 - 3: Read the section on `assert()` on the class web site and understand it fully. Copy, paste, and edit the code below the question.

Question 1: Rewrite `assert.cpp` to use `assert()` to ensure a negative square root is not taken.

```
/* Programmer:  Dylan McDonald
   Class & Section:  CS 54, Section <n>
   Date:  2 October 2007
   Purpose:  Make sure the user passed Algebra II */

#include <cmath>
#include <cassert> /* include this for the assert() function */
#include <iostream>
using namespace std;

int main()
{
    double answer, argument;

    /* ask the user what they want to square root */
    cout << "What number do you want to square root? ";
    cin >> argument;
    assert(argument >= 0);

    /* show the user how smart we are */
    answer = sqrt(argument);
    cout << "sqrt(" << argument << ") = " << answer << endl;
    return 0;
}
```

Question 2: Rewrite search.cpp to guarantee a positive searchPoint.

```
/* Programmer: Dylan McDonald
   Class & Section: CS 54, Section <n>
   Date: 2 October 2007
   Purpose: Randomly search a search space & get close to a solution */

#include <cstdlib>
#include <ctime>
#include <cassert> /* include this for the assert() function */
#include <iostream>
using namespace std;

int main()
{
    /* seeds the random number generator */
    srand(time(NULL));

    /* randomly decide where to start searching */
    int searchPoint = rand() - rand();
    int endPoint = searchPoint * rand() % 10;

    assert(searchPoint > 0);

    /* for a positive search point, randomly get closer */
    while (searchPoint < endPoint)
    {
        /* good enough for government work */
        if ((abs(searchPoint - endPoint)) < 42)
        {
            cout << "We're close enough!" << endl;
            break;
        }

        /* randomly move on */
        searchPoint += rand() % 100;
    }
    return 0;
}
```

Question 3: Write a function that divides the first input by the second input. Assert the function will not divide by zero.

```
double divide(double upstairs, double downstairs);

double divide(double upstairs, double downstairs)
{
    /* make sure the answer is valid & find it if it is */
    double answer;
    assert(downstairs != 0);
    answer = upstairs / downstairs;
    return answer;
}
```

Questions 4 & 5: Read the section on gdb on the class web site and fully understand it. Record all your results below the question.

Question 4: Set a break point at line 29 and continue through the while loop in `fall.cpp`. Record the values of `fallTime`, `acceleration`, and `currVelocity` at each step. Do this for four iterations.

Iteration	fallTime	acceleration	currVelocity
0	0	9.8066499999999994	0.98066500000000001
1	0.10000000000000001	9.7906164930169126	1.9597266493016914
2	0.20000000000000001	9.7426207859520417	2.9339887278968955
3	0.30000000000000004	9.6631327658480721	3.9003020044817029

Question 5: Set a break a point at the beginning of function `f` in `infiniteFunctions.cpp`. Step 5 - 10 times and do a backtrace. Paste the output of the backtrace here.

```
#0 g (x=7) at infiniteFunctions.cpp:26
#1 0x0804842b in f (x=6) at infiniteFunctions.cpp:20
#2 0x08048443 in g (x=5) at infiniteFunctions.cpp:26
#3 0x0804842b in f (x=4) at infiniteFunctions.cpp:20
#4 0x08048443 in g (x=3) at infiniteFunctions.cpp:26
#5 0x0804842b in f (x=2) at infiniteFunctions.cpp:20
#6 0x08048443 in g (x=1) at infiniteFunctions.cpp:26
#7 0x0804842b in f (x=0) at infiniteFunctions.cpp:20
#8 0x0804840e in main () at infiniteFunctions.cpp:13
```

Question 6 (Extra Credit): What includes are needed in `fall.cpp`?

None. There is no need to include `cmath` since all the math this function does is performed using standard arithmetic operators. There is no need for `iostream` since the program makes no attempt to print anything out to the screen or take input from the user. There is no need for functions from `cstdlib` or `ctime` (e.g., `rand()`, `abs()`, etc.), either. Therefore, nothing is needed. This raises the question: is the `using namespace std` statement needed? No, since nothing needs to be including from the standard namespace, then there is no need to use the standard namespace. So, we can leave that out, too. Thus, `fall.cpp` is fine how it is.