PROJECT 2, AVERAGE CASE COMPLEXITY AND MONTE CARLO

ALEXANDER WOOD

First I will print out the code used for the projects. A table of the requested results is located on the last page. The project `AverageCaseMonteCarlo_RealAverage` has the following code:

```
#include <iostream>
#include <cstdlib> // for random numbers
#include <cmath> // for math
#include <ctime> // for computing time

using namespace std;

// the upper bound on our random numbers, is changed for each experiment
int bound = 10000;

// return a uniformly distributed random number between
// 1 and some number, after the % sign.
int uniformRandom()
{
  return ( (int)(rand()%bound + 1) );
                /* rand()%k generates
                a random number between
                1 and k */
}

int main()
{
    const int n = 50; // Number of integers per list, is the columns of Sequence
    const int NumberOfTests = 10000; // Number of lists we test, is the rows of Sequence

    int hits = 0; // initialize the counter hits to 0

    int x = 0; // x is a random integer between 0 and bound
    srand(time(NULL));
    x = uniformRandom();

    cout << "A random integer x between 1 and " << bound << " is x=";
    cout << x << "." << endl;
    cout << "We are running the test on arrays of " << n << " integers ";
    cout << NumberOfTests << " times." << endl;
```

Date: October 5, 2015.

```
        /* Generate a random sequence of integers between 0 and
        bound using a random number generator. Save this new
        sequence in a two dimensional array Sequence[10000][50]

        If x is equal to any of the numbers generated in the
        sequence increment hits by one (if x appears on the same
        sequence then do not increment hits every time)

        Repeat these steps 10000 times. */

cout << endl;
cout << "The program now creates Sequence[10000][n] and fills it with random ";
cout<< "integers between 0 and " << bound << "." << endl;

int Sequence[NumberOfTests][n] = {0}; // create 10000 x n matrix

// assigns random values to entries of 10000 x n matrix
for(int ColumnAssign = 0; ColumnAssign < n; ++ColumnAssign)
{
    for(int RowAssign=0; RowAssign < NumberOfTests; ++RowAssign)
    {
        Sequence[RowAssign][ColumnAssign] = uniformRandom();
    }
}

// increments hits for each row which contains x
for(int Row = 0; Row < NumberOfTests; ++Row)
{
    for(int Column = 0; Column < n; ++Column)
    {
        if (Sequence[Row][Column] == x)
        {
            hits++;
            break;
        }
    }
    // After break you end up here
}

cout << "hits = " << hits << endl;

double doublehits = hits;
double doubleNumberOfTests = NumberOfTests;
double q = doublehits / doubleNumberOfTests;
```

```
cout << "The calculated average is A(n)= ";
cout << (( q /n )*(n*(n+1)/2)) + ((1-q)*n) << endl;

cout << "Next we want to find the real average." << endl;

    /* Using the sequences created, we will run algorithm Find
    10,000 times, each time using an input of Sequence[][]. Let
    total-steps be a variable that tells you the number of steps
    executed so far, and keep doing this until all 10,000 steps
    are executed.
    */

int totalSteps = 0; // the number of steps executed so far

// adds total number of steps in rows where x appears
for(int Row = 0; Row < NumberOfTests; ++Row)
{
    for(int Column = 0; Column < n; ++Column)
    {
        if (Sequence[Row][Column] == x)
        {
            totalSteps += Column + 1;

            break;
        }
    }
}


bool True = 0; // used in for loop to tell us if the row had x in it
               // we add n to totalSteps for each row which does not
               // contain x

// adds total number of steps in rows where x does not appear
for(int Row = 0; Row < NumberOfTests; ++Row)
{
    for(int Column = 0; Column < n; ++Column)
    {
        if (Sequence[Row][Column] == x)
        {
            True = 1;

            break;
        }

        True = 0;
```

```
        }

        if (True == 0)
        {
            totalSteps += n;
        }
    }

    double doubletotalSteps = totalSteps;

    cout << "total steps: " << totalSteps << endl;
    cout << "The real average A1(n) = " << doubletotalSteps / doubleNumberOfTests << endl;
}
```

Bound	Calculated Averate	Real Average
30	29.8585	24.7154
50	34.2416	32.059
80	38.6247	37.304
100	40.124	38.9988
1000	48.797	48.8728
10,000	49.902	49.8947
infinite	50	50