```cpp
#include <iomanip>
#include <limits.h> // need for infinity number
#include <ctime>
#include <iostream>
using namespace std;
//                    M O N T E  C A R L O
const int N = 50;
// functions declarations
void Calculation(int sequence[N], int
current_bound, int x);
int Find(int x, int A[], int N);

int main() {
    int sequence[N];
    int current_bound;

    int bound[7] =
{30,50,80,100,1000,10000,INT_MAX};
    //INT_MAX is a value that specifies max integer
value, it can not go beyond this value
    // top of the table
    cout << left << setw(15) << "BOUND" << setw(25)
<< "CALCULATED AVERAGE"
```

```
    << setw(20) << "REAL AVERAGE" << endl;
    cout <<
"=========================================================
=========" << endl;

    srand(time(NULL)); //random generator number

    //integer x that we are looking is between 0
and bound
    for (int i = 0; i < 7; i++) {
        int x = rand() % bound[i];
        current_bound = bound[i];
        Calculation(sequence, current_bound, x);
    }
}

// Check if x is in the sequence
int Find(int x, int A[], int N){ // array of N size
    int j;
        for (j = 0; j < N; j++)
            {
                if (x == A[j]) return (j + 1); //
position is offset by one from the index
```

```
            }
        return 0;} // x is not a member of the
array


void Calculation(int sequence[], int current_bound,
int x)
{
    double average_case = 0;
    double real_average = 0;
    double total_steps = 0;
    double q = 0;
    double hits = 0;
    double search_result = 0;

    // loop to count average
    for (int j = 0; j < 10000; j++)
    {
        for (int i = 0; i < N; i++)
        {
            sequence[i] = rand() % current_bound;
                if (x == sequence[i])
```

```
                    {
                        hits++;
                    }
            }

            search_result = Find(x, sequence, N);

            if (search_result == 0)
                {// If x is not located, include 50
steps.
                    total_steps += 50;
                }
                else if (search_result != 0)
                {// if x is found, add steps to
search_result
                     total_steps += search_result;
                }// Increment the step count by 1
                total_steps++;
        }

        q = hits / 10000;
```

```
    average_case = (N - ((N * q) / 2) + (q / 2)); // average case calculate

    real_average = total_steps / 10000; // real average calculation

    // results
    cout << fixed << setprecision(1) << left << setw(15) << current_bound
    << setw(25) << average_case << setw(20) << real_average << endl;

}
```