

CS377 – Fall 2007

Homework 6

Due Date: December 4, 2007 before 12:30pm

Your assignment is to demonstrate how a simple binary search routine is tested to ensure the desired test coverage criteria. Two implementations of the routine are supplied with this assignment, one in Java and the other in C++. The routine contains errors your tests should help you discover. You may populate the sorted sequence of integers used in your search in any way you desire. You are required to test binary search both as a white-box and a black-box module. You must perform the following:

1. As a white-box, you must devise a minimal set of test cases to ensure branch coverage of the routine. Discuss whether branch coverage testing helped you discover the errors in the routine. If so, how? If not, why do you think it did not?
2. As a black box, you must select a minimal set of test cases to thoroughly test the routine, including all boundary conditions. Discuss whether black-box testing helped you discover the errors in the routine. If so, how? If not, why do you think it did not?

As your deliverables, you must turn in a list and explanation of your test cases: why have you selected them, what do they accomplish, and why do you think they are minimal, but sufficient to ensure the above criteria? In addition, you should discuss the above set of questions in the context of white-box and black-box testing, respectively. You should turn in *a script* showing all of your runs, as well as any modifications you have made to either the Java or the C++ code.

SUBMISSION INSTRUCTIONS

You are required to submit all your files electronically to prakashg@usc.edu and nenos@usc.edu in a single zip file named

CS377-HW6-<yourfullname>.zip

A failure to follow this submission requirement will result in a 10% deduction from your score.

A sample Java version of the binary search routine is given below.

```
import java.util.*;

public class Search {

    public int BinarySearch(Vector v, int key)
    {
        if (v == null)
            throw new IllegalArgumentException("null arg");

        int low = 0;
        int high = v.size() - 1;
        while (low <= high)
        {
            int mid = (low + high) / 2;
            int v_element = ((Integer)v.elementAt(mid)).intValue();
            if (key <= v_element)
                high = mid - 1;
            else if (key >= v_element)
                low = mid + 1;
            else
                return mid;
        }
        return -1;
    }

    public static void main(String args[])
    {
        Search srch = new Search();
        Vector v = new Vector();
        int N = 10000;
        int i = 0;

        //populate the vector however you want
        for (i = 0; i < N; i++)
            v.addElement(new Integer(i));

        //search for whatever element you want
        for (i = 0; i < N; i++)
        {
            int result = srch.BinarySearch(v,i);
            //you may print the result here
        }
    }
}
```

A sample C++ version of the binary search routine is given below.

```
#include <iostream.h>
#include "vector.h"

const int NOT_FOUND = -1;

/*
 * Compiling instruction:
 *   g++ BinarySearch.ccs
 *   The executable is a.out.
 *
 * Returns index where item is found or -1 if not found
 */

int binarySearch( int * array,
                 int size, // size of the array
                 int key //key
                 )
{
    int low = 0, high = size - 1;

    while( low <= high )
    {
        int mid = ( low + high ) / 2;

        if( array[ mid ] <= key )
            low = mid + 1;
        else if( array[ mid ] >= key )
            high = mid - 1;
        else
            return mid;    // Found
    }
    return NOT_FOUND;    // NOT_FOUND is defined as -1
}

int main( )
{
    const int SIZE = 8;
    int array[SIZE];

    cout<<"The array is:"<<endl;
    for( int i = 0; i < SIZE; i++ ) {
        array[ i ] = i * 2;
        cout<<array[i];
    }

    cout<<endl;

    for( int j = 0; j < SIZE * 2; j++ )
        cout << "Found " << j << " at " << binarySearch( array, SIZE, j )
<<endl;
    return 0;
}
```