CSCI 3901

Lab 5 – Report

Team Members:
Benny Daniel Tharigopala [B00899629]
Nadipineni Hemanth Kumar [B00899473]

Questions:

1. We usually want you to re-use existing code and infrastructure whenever possible. Why might you create your own exception?

   Answer:

   Although I would reuse existing code, I might be developing an additional functionality or enhancing an existing one. In order to handle inadvertent situations that I might have not anticipated and to present errors to users in a meaningful manner, I might create my own exceptions.

2. We added parameters to the Fibonacci method. However, those parameters aren't very meaningful to a general user. What would you do to the code to make it more accessible for a general user?

   Answer:

   I would print what the parameters signify, to standard output and articulate their relevance.

   I would also add comments beside the parameters.

3. How would you recommend for someone to develop a loop invariant?

Answer:

We recommend developing a loop variant (especially in iterative programs) that satisfies all three properties i.e., Initialization, Maintenance, Termination. It means that the loop variant:

1) should be true before first execution of loop [Establishment]

2) should be true after the iteration if it was before the iteration [Preservation]

2.1) the loop should be achieving what it wants if the loop condition is false (loop will exit) but the invariant holds [Postcondition]

3) should tell us something helpful after the termination [Termination]

For example,

In our binaySearch():

The invariant is the logical statement that check the

1)Here, we have three parts of establishment

· Array is sorted as it is the precondition

· As r = n-1 (n = arr.length) on line 40 and r>=l on line 11, we know that are looping towards l->r

· ‘x’ is in arr[l,….,r] as that is the entire loop

2)In our entire program, you may notice that the array never changed. So, this means the invariant is preserved.

· Let’s consider ‘x’ is in our array. When arr[m]>x, we pass m-1 as ‘r’ so that it is validated that x belongs to arr[l,..m-1]. This way we are not modifying the array and it is still preserved.

· Let’s consider ‘x’ is not in our array. When the first condition is false, it goes as arr[m+1,…,r] since the array is already sorted and ‘x’ is not in the required range. We now know that the array isn’t altered for this condition too.

3)The invariant stated that the array is sorted after the loop is terminated and also it checked full loop for ‘x’.

4. How can loop invariants help you in programming, even if you don’t include them directly as assertions in your code?

Answer:

Without using assertions, we can check for the conditions which satisfy Preservation, Postcondition and Termination.

Using our binarySearch() example:

1)’arr’ must be sorted in ascending order. It cannot satisfy Preservation as we can’t say that there will be ‘x’ in the updated array i.e,...arr[l,….,r].

2)’l’ must be less than or equal to ‘r’. We cannot say if we are going in the right direction after splitting ‘m’ on line 11.

3)(arr[m]==x), Without this expression, we aren’t sure if the loop has found anything. This way we make sure that postcondition doesn’t fail.