UMassAmherst | Manning College of Information & Computer Sciences

Programming Methodology

Lab 11: Program Correctness

Wednesday, April 24th, 2024

Weekly Lab Agenda

- Go over reminders/goals
- Review past material
- Work in groups of 2-3 to solve a few exercises
    - Lab leaders will assign new groups this week
- Discussion leaders will walk around and answer questions
- Solutions to exercises will be reviewed as a class
- Attendance taken at the end

Reminders

- You should be working on your group project.
- Have a planned timeline to avoid last minute rush!
- Reach out to us if there is any issues in your team.
  - Private post on campuswire or
  - Email to mkuechen@umass.edu

UMassAmherst | Manning College of Information & Computer Sciences

Today’s Goals

- Practice working on program correctness.

Question 1

Please write your solution to question 1 on paper. You will submit your work to gradescope before we review the solution, and we'll grade your work to give you feedback.

UMassAmherst | Manning College of Information & Computer Sciences

Question 1

What is the largest value A for which the loop below terminates with n=20?

Use an invariant to justify your answer.

#Fall 2022 Final Exam

```
let n: number = 0;

let s: number = A;

while (s <= 100) {

          if (n % 2 > 0) {

                  s += n;

         } else {

         }

         n = n + 1;

}
```

Question 1

Please submit to gradescope now before we go over the solution. We'll grade these to give you feedback, but the score you receive will not impact your grade. Your grade for this lab is as always based on attendance only.

Solution

UMassAmherst | Manning College of Information & Computer Sciences

```
let n: number = 0;
let s: number = A;
 // s = A + oddsum(n)          // sum of odd numbers less than n
while (s <= 100) {
        // s = A + oddsum(n)
        if (n % 2 > 0) {

                s += n;

        } else {

        }

n = n + 1;


}
```

The loop adds odd numbers.
Our invariant is that s is A (initial value) plus the sum of odd numbers less than n.

Solution

UMassAmherst | Manning College of Information & Computer Sciences

```
let n: number = 0;
let s: number = A;
 // s = A + oddsum(n)              // sum of odd numbers less than n
while (s <= 100) {
        // s = A + oddsum(n)
        if (n % 2 > 0) {
                // s = A + oddsum(n)
                s += n;

        } else {

        }

n = n + 1;


}
```

The loop adds odd numbers.
Our invariant is that s is A (initial value) plus the sum of odd numbers less than n.

Solution

UMassAmherst | Manning College of Information & Computer Sciences

```
let n: number = 0;
let s: number = A;
 // s = A + oddsum(n)              // sum of odd numbers less than n
while (s <= 100) {
          // s = A + oddsum(n)
          if (n % 2 > 0) {
                    // s = A + oddsum(n) && n % 2 > 0
                    s += n;
                    // s = A + oddsum(n) + n && n % 2 > 0
                    // s = A + oddsum(n+1)
          } else {

          }

n = n + 1;


}
```

We have added n, which is odd, so we have the sum of all numbers less than n+1

Solution

UMassAmherst | Manning College of Information & Computer Sciences

```
let n: number = 0;
let s: number = A;
 // s = A + oddsum(n)              // sum of odd numbers less than n
while (s <= 100) {
          // s = A + oddsum(n)
          if (n % 2 > 0) {

                    s += n;

          } else {
                    // s = A + oddsum(n)  && n % 2 === 0
                    // s = A + oddsum(n + 1)
          }

n = n + 1;


}
```

For the else block since n is even, oddsum(n) = oddsum(n+1)

Solution

UMassAmherst | Manning College of Information & Computer Sciences

```
let n: number = 0;
let s: number = A;
 // s = A + oddsum(n)              // sum of odd numbers less than n
while (s <= 100) {
          // s = A + oddsum(n)
          if (n % 2 > 0) {

                    s += n;
                    // s = A + oddsum(n+1)
          } else {
                    // s = A + oddsum(n+1)
          }
         // s = A + oddsum(n+1) (common for both branches)
         n = n + 1;


}
```

We have a common form after the if statement.

Solution

UMassAmherst | Manning College of Information & Computer Sciences

```
let n: number = 0;
let s: number = A;
 // s = A + oddsum(n)              // sum of odd numbers less than n
while (s <= 100) {
        // s = A + oddsum(n)
        if (n % 2 > 0) {

                s += n;

        } else {

        }
        // s = A + oddsum(n+1) (common for both branches)
       n = n + 1;
       // s = A + oddsum(n)

}
```

The assignment rule for n= n + 1 restores the invariant.

Solution

UMassAmherst | Manning College of Information & Computer Sciences

```
let n: number = 0;
let s: number = A;
 // s = A + oddsum(n)              // sum of odd numbers less than n
while (s <= 100) {
        // s = A + oddsum(n) && s <= 100
        if (n % 2 > 0) {

                s += n;

        } else {

        }
        n = n + 1;
       // s = A + oddsum(n)
}
// n = 20 & A + oddsum(n-1) <= 100 && A + oddsum(n) > 100
```

To find A, we impose:
s <= 100 for n=19 (enter loop)
s > 100 for n=20 (exit loop)

We have oddsum(20) = 1+3+..+19 = 10*(1+19)/2 = 100

A + (100-19) <= 100 && A+100 > 100
0 < A <= 19

The largest value is 19.

Question 2

UMassAmherst | Manning College of Information & Computer Sciences

In class yesterday we saw the following code, which finds the index of the biggest element of array arr between indexes a and b, inclusive.

```
function biggest(arr, a, b) {
  let big = a;
  for (let i=a+1; i < b; i=i+1) {
    // big: the index of the biggest element in arr[a..i-1]
    if (arr[i] > arr[big]) {
      big = i;
      // big is the index of the biggest element in arr[a..i]
    }
    else {
      // big is the index of the biggest element in arr[a..i-1]
      // and arr[i] is not bigger than arr[big]
      // therefore big is the index of the biggest element in arr[a..i]
    }
    // "implied i = i+1"
    // big is the index of the biggest element in arr[a..i-1]
  }
  return big;
}
```

Some assertions are missing from the code listed here, but we saw that the invariant for variable big is maintained.

Question 2

UMassAmherst | Manning College of Information & Computer Sciences

In class yesterday we saw the following code, which finds the index of the biggest element of array arr between indexes a and b, inclusive.

```
function biggest(arr, a, b) {
  let big = a;
  for (let i=a+1; i < b; i=i+1) {
    // big: the index of the biggest element in arr[a..i-1]
    if (arr[i] > arr[big]) {
      big = i;
      // big is the index of the biggest element in arr[a..i]
    }
    else {
      // big is the index of the biggest element in arr[a..i-1]
      // and arr[i] is not bigger than arr[big]
      // therefore big is the index of the biggest element in arr[a..i]
    }
    // "implied i = i+1"
    // big is the index of the biggest element in arr[a..i-1]
  }
  return big;
}
```

Let's take a look at the stopping criteria of this loop, to see if it's the right one.

Question 2

UMassAmherst | Manning College of Information & Computer Sciences

In class yesterday we saw the following code, which finds the index of the biggest element of array arr between indexes a and b, inclusive.

```
function biggest(arr, a, b) {
  let big = a;
  for (let i=a+1; i < b; i=i+1) {
    // big: the index of the biggest element in arr[a..i-1]
    if (arr[i] > arr[big]) {
      big = i;
      // big is the index of the biggest element in arr[a..i]
    }
    else {
      // big is the index of the biggest element in arr[a..i-1]
      // and arr[i] is not bigger than arr[big]
      // therefore big is the index of the biggest element in arr[a..i]
    }
    // "implied i = i+1"
    // big is the index of the biggest element in arr[a..i-1]
  }
  return big;
}
```

If big is the index of the biggest element in arr[a..i-1], what value do we want i-1 to have when we exit the loop?

Question 2

UMassAmherst | Manning College of Information & Computer Sciences

If big is the index of the biggest element in arr[a..i-1], what value do we want i-1 to have when we exit the loop?

Question 2

UMassAmherst | Manning College of Information & Computer Sciences

If big is the index of the biggest element in arr[a..i-1], what value do we want i-1 to have when we exit the loop?

answer: We want i-1 to be equal to b, so that, combined with the invariant for big, we can say that big is the index of the biggest element in arr[a..b]

Question 2

UMassAmherst | Manning College of Information & Computer Sciences

We want i-1 to be equal to b, so that, combined with the invariant for big, we can say that big is the index of the biggest element in arr[a..b]

```
function biggest(arr, a, b) {
  let big = a;
  for (let i=a+1; i < b; i=i+1) {
  …
  }
  return big;
}
```

With this driver, what will be the relationship between i and b once we exit the loop?

Question 2

UMassAmherst | Manning College of Information & Computer Sciences

We want i-1 to be equal to b, so that, combined with the invariant for big, we can say that big is the index of the biggest element in arr[a..b]

```
function biggest(arr, a, b) {
  let big = a;
  for (let i=a+1; i < b; i=i+1) {
  …
  }
  return big;
}
```

With this driver, what will be the relationship between i and b once we exit the loop?

Answer: i >= b, the negation of the driver.

Question 2

UMassAmherst | Manning College of Information & Computer Sciences

With this driver, what will be the relationship between i and b once we exit the loop?

Answer: i >= b, the negation of the driver.

If i>b, then we're OK, because, since they are both integers, then we have i-1>=b

But if i===b, then we have a problem, because our invariant for big is big is the index of the biggest element in arr[a..i-1], which would mean that big is the index of the biggest element in arr[a..b-1].

YIKES! We have the wrong driver!

Well, that's OK, though. Our analysis detected it, even before testing. Now we just have to fix it.

Question 2

UMassAmherst | Manning College of Information & Computer Sciences

big is the index of the biggest element in arr[a..b-1].

YIKES! We have the wrong driver!

Use the invariant/assertions/formal method of proving things, as opposed to trial, testing, and editing to determine the correct termination condition for our loop.

Question 2

UMassAmherst | Manning College of Information & Computer Sciences

big is the index of the biggest element in arr[a..b-1].

YIKES! We have the wrong driver!

Use the invariant/assertions/formal method of proving things, as opposed to trial, testing, and editing to determine the correct termination condition for our loop.

After you determine the correct loop driver, write a test that verifies that the previous driver was incorrect, but the new one is correct. Give an example.