Midterm I Exam Review

15-122 Principles of Imperative Computation
Alex Cappiello

October 7, 2012

1 Modular Arithmetic

In C0, values of type `int` are defined to have 32 bits. In this problem we work with a version of C0 where values of type `int` are defined to have only 9 bits. In other respects it is the same as C0. All integer operations are still in two's complement arithmetic, but now modulo 2^9. All bitwise operations are still bitwise, except on only 9 bit words instead of 32 bit words.

Task 1. Fill in the missing quantities, in the specified notation.

a. The minimal negative integer, in decimal: _____

b. The maximal positive integer, in decimal: _____

c. -4, in hexadecimal: 0x_____

d. 44, in hexadecimal: 0x_____

e. `0x49`, in decimal: _____

Task 2. Assume `int x` and `int y` have been declared and initialized to unknown values. For each of the following, indicate if the expression always evaluates to `true`, or if it could sometimes be `false`. In the latter case, indicate a counterexample in the C0 dialect described here by giving a value for `x` and `y` that falsifies the claim. You may use decimal or hexadecimal notation. You may use the functions `int_min()` and `int_max()` and assume they are correct for 9 bit ints.

a. `x >= x - 1` _____

b. `(~x) ^ (x) == -1` _____

c. `x+(y+1)-2*(x-1)-3 == -x+y` _____

d. `(x!=-x || y!=-y) || x==y` _____

e. `((x<<1)>>1) | (x & 0x80) == x` _____

f. `(((x>>1)<<1) | (x&1)) == x` _____

g. `x <= (1<<(7-1))-1` _____

h. `x+x == 2*x` _____

2 Ternary Search.

Consider the following variation of binary search algorithm. Instead of checking the middle element of the sorted array $A[]$, check the element at position $n/3$. Then proceed in the same way as in binary search. If you are looking for x then

- if $x == A[n/3]$ you have found it.
- if $x < A[n/3]$ you search the first third of the array, namely at indexes $< n/3$.
- if $x > A[n/3]$ you search the rest two thirds of the array at indexes $> n/3$.

Consider the following implementation:

```
int tersearch(int x, int[] A, int n)
//@requires 0 <= n && n <= \length(A);
//@requires is_sorted(A, n);
/*@ensures (-1 == \result && !is_in(x, A, n))
        || ((0 <= \result && \result < n) && A[\result] == x);
  @*/
{
  int lower = 0;
  int upper = n;
  while (lower < upper)
    //@loop_invariant 0 <= lower && lower <= upper && upper <= n;
    //@loop_invariant lower == 0 || A[lower-1] < x;
    //@loop_invariant upper == n || A[upper] > x;
    {
      int mid = lower + (upper-lower)/3;
      if (A[mid] < x)
        lower = mid+1;
      else if (A[mid] > x)
        upper = mid;
      else //@assert A[mid] == x;
        return mid;
    }
  //@assert lower == upper;
  return -1;
}
```

Task 1. Prove the correctness of this function, using the method presented in class.

Task 2. Find an expression for the worst-case runtime of ternary search. Compare this asymptotically to binary search.

3 Runtime Analysis

Determine the big O complexity of the following expression:

$$T(n) = 4n^5 + 10n^3 + \log n$$

Use the formal definition of big O to justify your answer.