

Name:

Exam 1, CS 201: There are five problems on this test. You do not need to calculate final numbers, answers in the form of $15! + 2^4$ are acceptable. Partial credit depends on clear explanations of your work. Make sure you proofread your answers. Good Luck.

Facts which are true but may or may not be relevant:

- Program Correctness Proofs have 6 parts. (1) State loop invariant. (2) Prove loop invariant is true before the loop. (3) Prove loop invariant is true after an arbitrary iteration of the loop. (4) Prove the loop invariant and the loop stopping condition imply the final assertion. (5) Prove that the loop terminates.
- $\lfloor x \rfloor$ is the largest integer less than or equal to x .
- Q is the set of rational numbers, Z is the set of integers, R is the set of real numbers.
- $\text{sign}(x) = 1$ if $x \geq 0$, $\text{sign}(x) = -1$ if $x < 0$
- $|x| = x$ if $x \geq 0$, $|x| = -x$ if $x < 0$
- Some induction proofs require more than one base case.
- Given sets A, B , an item is in the set $A - B$ if and only if (it is in set A and it is not in set B).
- Given sets A, B , an item is in the set $A \cup B$ if and only if (it is in set A or it is in set B).
- Two sets A, B are equal when an element x is in A if and only if x is in B .

(for graders use:)

1	2	3	4	5	Total
---	---	---	---	---	-------

1) (8 points) For this problem, a poker hand is DEFINED to be an unordered set of 5 cards, drawn (without replacement) from the 52 total cards in a deck (13 ranks, and 4 suits).

(a) A straight is any poker hand that has 5 consecutive numbers, for instance, the 8, 9, 10, Jack, Queen, but they can be different suits. How many possible poker hands are a “straight”?

The answer (assuming that an ace can count at low or high), is

10×4^5 , which is 10 choices for the which rank (the “number part of the card, instead of the suit”) can be the first card of the straight, and then for each of the five cards, there are 4 choices for which suit.

(b) A flush is a poker hand where every card has the same suit for in-

stance the 2 of clubs, 4 of clubs, 8 of clubs, Jack of clubs, Ace of clubs. How many possible poker hands are a “flush”?

The answer for this can be computed as:

$4 \times (13\text{choose}5)$, because to make the hand you have to pick one of 4 suits, and then you have to pick 5 out of 13 ranks.

2) (8 points) For each function f defined below, say whether f is (a) 1-1, and (b) onto. When f is not one of those, argue, briefly, why. Pay attention to the domain and co-domain specified.

(a) $f : R \longrightarrow Z, f(x) = \lfloor 2x \rfloor$

- 1-1? (no, $f(1.1) = f(1.2)$)
- onto? (yes)

(b) $f : Z \longrightarrow Z, f(x) = 2^x - x^2$

This function creates values not in the co-domain. if $x = -1$, then $f(x) = 1/2 - 1 = -1/2$, which is not an element of Z . even if we ignore this problem, the answers to the following are

- 1-1? no,
- onto? no.

(c) $f : Z \times Z \longrightarrow Q, f(x, y) = \frac{x}{y}$

- 1-1? (no, $f(2,2) = f(3,3)$)
- onto? (yes)

(d) $f : Z \longrightarrow R, f(x) = \frac{1}{x+0.5}$

- 1-1? (yes)
- onto? (no, there is no x such that $f(x) = 1$).

3) (8 points) Use a loop invariant to prove the correctness of the following algorithm to compute the maximum of a sequence of integers $a_0, a_1, a_2, \dots, a_{n-1}$ with respect to the initial assertion ($n > 0$) and the final assertion that $ans = \max(a_0, a_1, a_2, \dots, a_{n-1})$.

```

procedure computeMax( $a_0, a_1, a_2, \dots, a_{n-1}$ )

     $ans = a_0$ ;
     $i = 1$ ;
    while ( $i < n$ ) do
        if  $a_i > ans$  then
             $ans = a_i$ 
        fi
         $i = i + 1$ 
    od
    return  $ans$ 

```

initial assertion is that we get a list, of, like, numbers. the final assertion is that $ans = \max(a_0, \dots, a_{n-1})$.

Ok. Our loop invariant is

$L = \text{“}ans = \max(a_0, \dots, a_{i-1})\text{”}$

Now, lets prove the first two lines make the loop invariant true:

$\{ans = a_0, i = 1\}$ L:

after the above code lines, ans is defined to be a_0 , so $ans = \max(a_0)$.

Next lets prove this remains true during the loop:

L { loop code } L

First case: $a_i > ans$ code sets: $ans' = a_i, i' = i + 1$

by L, $ans = \max(a_0, \dots, a_{i-1})$

but if $a_i > ans$, then a_i is the max of (a_0, \dots, a_i) .

so after loop runs $ans' = \max(a_0, \dots, a_i) = \max(a_0, \dots, a_{i'-1})$.

Second case: $a_i \leq ans$ code sets: $i' = i + 1$

by L, $ans = \max(a_0, \dots, a_{i-1})$

but if $a_i \leq ans$, then ans is already the max of (a_0, \dots, a_i) .

so after loop runs $ans = \max(a_0, \dots, a_i) = \max(a_0, \dots, a_{i'-1})$.

Finally, when the loop ends, $i = n$. L states $ans = \max(a_0, a_{i-1})$, and

if we substitute $i = n$, then we get $ans = \max(a_0, a_{n-1})$, which is our final assertion.

Finally, since i increases by one each time through the loop, the program terminates.

4) (8 points)

Define a function $g(n)$ for integer $n \geq 0$ by the following recurrence:

$$g(0) = 3$$

$$g(1) = 4$$

$$\text{for } n \geq 2, g(n) = 4g(n-1) - 4g(n-2).$$

(a, 2 points) Calculate: $g(2)$, $g(3)$, $g(4)$, $g(5)$

$$g(0) = 3$$

$$g(1) = 4$$

$$g(2) = 4$$

$$g(3) = 0$$

$$g(4) = -16$$

$$g(5) = -64$$

(b, 6 points) Prove that $g(n) = (3-n)2^n$ for $n \geq 0$.

Lets prove two base cases:

$$g(0) = 3 = (3-0)2^0 = 3 \times 2^0 = 3 \times 1 = 3$$

$$g(1) = 4 = (3-1)2^1 = 2 \times 2^1 = 4 \times 1 = 4$$

Lets define $P(k) = "g(k) = (3-k)2^k"$.

Lets assume $P(k-1)$, $P(k)$ and prove $P(k+1)$.

$$P(k+1) = "g(k+1) = (3-(k+1))2^{k+1}" . g(k+1) = 4(g(k)) - 4(g(k-1)).$$

By assumption, this is:

$$= 4(3-k)2^k - 4(3-(k-1))2^{k-1}$$

$$= 2(3-k)2^{k+1} - (3-(k-1))2^{k+1}$$

$$= (2(3-k) - (3-(k-1)))2^{k+1}$$

$$= (6-2k - (4-k))2^{k+1}$$

$$= (2-k)2^{k+1}$$

$$= (3-(k+1))2^{k+1}$$

which proves the induction step.

