Computer Science Foundation Exam

May 6, 2016

Section I A

COMPUTER SCIENCE

NO books, notes, or calculators may be used,
and you must work entirely on your own.

SOLUTION

Question #	Max Pts	Category	Passing	Score
1	10	DSN	7	
2	10	ANL	7	
3	10	ALG	7	
4	10	ALG	7	
5	10	ALG	7	
TOTAL	50		35	

You must do all 5 problems in this section of the exam.

Problems will be graded based on the completeness of the solution steps and <u>not</u> graded based on the answer alone. Credit cannot be given unless all work is shown and is readable. Be complete, yet concise, and above all <u>be neat</u>.

1) (10 pts) DSN (Recursive Functions)

Complete the function below, to create a ***recursive*** function ***productDigits*** that takes in 1 parameter, a non-negative integer, *n*, and returns the product of *n*'s digits. For example *productDigits(232)* will return 12 (2 x 3 x 2), and *productDigits(13999019)* will return 0.

```
int productDigits(int number){

   if(number < 10)
     return number;
   return (number%10) * productDigits(number/10);
}
```

Grading:

3 pts for base case
7 pts for recursive call - 1 pt return, 2 pts for getting last digit, 1 pt mult, 1 pt rec call
2 pts number/10

2) (10 pts) ANL (Summations)

a) (5 pts) Determine the value of the following summation, in terms of *n*: $\sum_{i=1}^{2n}(4i+7)$. Express your final answer as a polynomial in the form $an^2 + bn$, where a and b are integers.

$$\sum_{i=1}^{2n}(4i+7) = \left(4\sum_{i=1}^{2n} i\right) + \sum_{i=1}^{2n} 7$$
$$= 4 \times \frac{2n(2n+1)}{2} + 7(2n)$$
$$= 2(2n)(2n+1) + 14n$$
$$= 8n^2 + 4n + 14n$$
$$= 8n^2 + 18n$$

Grading: 1 pt for split, 2 pts for sum to i formula, 1 pt sum constant, 1 pt simplify

b) (5 pts) Determine the value of the summation below:

$\sum_{i=21}^{100}(3i+1) = \sum_{i=1}^{100}(3i+1) - \sum_{i=1}^{20}(3i+1)$

$= \sum_{i=1}^{100}(3i) + \sum_{i=1}^{100} 1 - (\sum_{i=1}^{20}(3i) + \sum_{i=1}^{20} 1)$

$= 3 \times \frac{100\times101}{2} + 100 - (3 \times \frac{20\times21}{2} + 20)$

$= 3 \times 50 \times 101 + 100 - 3 \times 10 \times 21 - 20$

$= 3 \times 5050 + 100 - 30 \times 21 - 20$

$= 15150 + 100 - 630 - 20$

$= 14600$

Grading (5 pts total): 1 pt for splitting sum, 3 pts for properly plugging into formulas for both sums, 1 pt for simplification

3) (10 pts) ALG (Stacks)

A stack of ***positive integers*** is implemented using the struct shown below. Using this implementation of the stack write the ***push*** and ***peek*** functions. ***Assume that when a struct stack is empty, its top variable is equal to -1.***

```
#define MAX 12

struct stack{
   int top;   /* indicates index of top */
   int nodes[MAX] ;
};

// Attempts to push value onto the stack pointed to by s.
// If the stack is full 0 is returned and no action is taken.
// Otherwise, value is pushed onto the stack and 1 is returned.
int push(struct stack* s, int value){

    if(s->top >= MAX-1)
        return 0;

    s->nodes[s->top + 1] = value;
    s->top++;
    return 1;
}
```

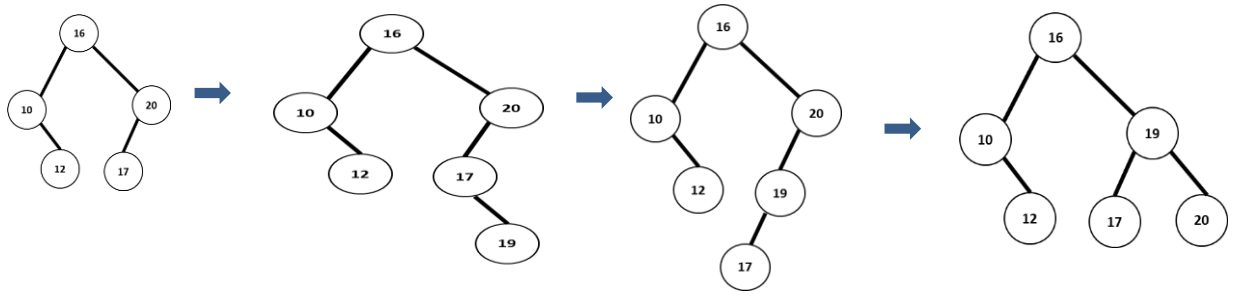
Grading: 2 pts for full case, 2 pts for insertion, 1 pt update top, 1 pt return

```
// Returns the value at the top of the stack. If the stack is
// empty, -1 is returned.
int peek(struct stack* s){

    if(s-> top == -1)
       return -1;

    return s->nodes[s->top];
}
```

Grading: 2pt for empty case, 2 pts for return in regular case.

4) (10 pts) ALG (Binary Search Trees and Hash Tables)

a) (5 pts) Show the AVL tree created when 19 is added to the AVL tree below



Grading: 5 pts total - GIVE FULL CREDIT IF FINAL TREE IS CORRECT
1 for inserting 19 in the correct position
2 pt for left rotation
2 pt for right rotation (for rotation trace)

b) (5 pts) In a binary heap of 100 elements, how many elements are at a depth of 6 (lowest level) from the root of the heap? (Note: the depth of an element is the number of links that have to be traversed from the root of the tree to reach it.)

A binary heap fills in each row before moving onto the next, since its structure is always a complete binary tree. Thus, the number of nodes at depths 0 through 5 are 1, 2, 4, 8, 16, and 32, respectively. This sums to 63. Thus, the next **37 nodes** will all be at a depth of 6 from the root.

37

Grading: 2 pts for observation of heap structure, 2 pts for counting up the nodes at all the previous levels, 1 pt for calculating the final answer.

5) (10 pts) ALG (Base Conversion)

a) Convert the hexadecimal number AF2E9 to binary without first converting to the base 10 equivalent

A F 2 E 9

10 15 2 14 9 1010 1111 0010 1110 1001

1010 1111 0010 1110 1001

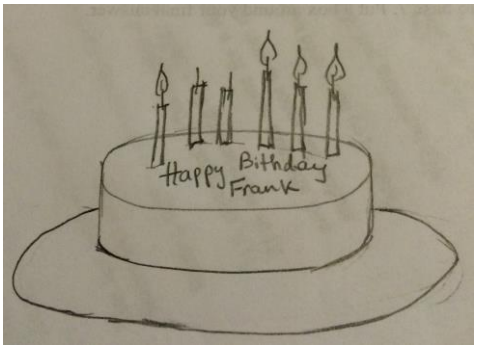
Grading: 5 pts total - 1 pt for each group of 4 bits. All 4 bits in the group have to be correct to get the point.

b) Frank is the team-lead for the software testing team at his job. He is celebrating his birthday. Some of his co-workers have baked a cake for the celebration and thought that it would be really cool to put candles on his cake to represent his age in binary. An unlit candle represents the 0 bit. From the pic of the cake below, how old is Max?

100111

32 4 2 1

= 32 + 4 + 2 + 1 = 39



Grading: 5 pts total
4 pts : 1 for decimal for each digit
1 pt for final answer

Note: Also give full credit for 32 + 16 + 8 + 1 = 57, though most students will read left to right.

Computer Science Foundation Exam

May 6, 2016

Section I B

COMPUTER SCIENCE

NO books, notes, or calculators may be used,
and you must work entirely on your own.

SOLUTION

Question #	Max Pts	Category	Passing	Score
1	10	ANL	7	
2	10	ANL	7	
3	10	DSN	7	
4	10	DSN	7	
5	10	ALG	7	
TOTAL	50		35	

You must do all 5 problems in this section of the exam.

Problems will be graded based on the completeness of the solution steps and <u>not</u> graded based on the answer alone. Credit cannot be given unless all work is shown and is readable. Be complete, yet concise, and above all <u>be neat</u>.

1) (10pts) ANL (Algorithm Analysis)

Determine the ***best case*** run time in terms *n* for each of the following functions/operations.

a) Finding the maximum value in an unsorted linked list of *n* elements O(n)

b) Inserting an item into a binary search tree of *n* elements O(1)

c) Inserting an item into a binary heap of *n* elements O(1)

d) Sorting an array of *n* elements using Merge Sort O(nlgn)

e) Deleting an element from a circular linked list of *n* elements O(1)

Grading: 1 pt each, must be correct to earn the point.

Determine the ***worst case*** run time in terms of *n* for each of the following functions/operations.

f) Deleting an item from an AVL tree of *n* elements O(lg n)

g) Deleting the minimum item from a binary min heap of *n* elements O(lg n)

h) Inserting an item into a binary search tree of *n* elements O(n)

i) Sorting an array of *n* elements using Heap Sort O(nlgn)

j) Deleting an element from a doubly linked list of *n* elements O(n)

Grading: 1 pt each, must be correct to earn the point.

2) (10 pts) ANL (Algorithm Analysis)

a) (5 pts) Given that a function has time complexity $O(n^2)$, if the function takes 338 ms for an input of size 13000, how long will the same function take for an input of size 8000?

$T(n) = cn^2$
$T(13000) = c13000^2 = 338ms$

$c = \frac{338}{13^2} \times \frac{1}{10^6} ms = \frac{338}{169} \times 10^{-6} ms = 2 \times 10^{-6} ms$

$T(8000) = c(8000)^2 = (2 \times 10^{-6} ms) \times 8^2 \times 10^6 = 128ms$

128 ms

Grading: 2 pts for computing the constant c, 2 pts substituting 8 into general equation to find time for input size of 8000, 1 pt for simplifying the final answer.

b) (5 pts) What is the run-time of the segment of code below, in terms of the variables *n* and *k*? Please provide a Big-Oh bound and briefly justify your answer. (Assume k has already been defined as set to a value prior to the code segment shown.)

```
int i, total = 0;
for (i=0; i<n; i+=2) {
    int start = k;
    while (start > 0) {
        total += ((k|i) & start);
        start /= 2;
    }
}
```

The outer loop runs n/2 times. The inner loop will always run O(lg k) times, since k never changes during the code segment and each iteration of the while loop divides start by 2. Since the two loops are independent of each other in terms of number of times they run, it follows that the run time of the code segment is O(nlg k).

O(nlg k)

Grading: 2 pts outer loop analysis, 2 pts inner loop analysis, 1 pt final answer

3) (10 pts) DSN (Linked Lists)

Write a function, `mode`, that takes in a pointer to the front of a linked list storing integers, and returns the mode of the list of integers. Recall that the mode of a list of values is the value that occurs most frequently. You may assume that all of the integers in the list are in between 0 and 999, inclusive. If there is more than one mode, your function must return the smallest of all of the modes. (For example, if the list contains the values 2, 4, 3, 2, 2, 4, 1, and 4, your function must return 2 and should NOT return 4, since both 2 and 4 occur three times in the list but 2 is smaller than 4.) Hint: declare an auxiliary array inside of the mode function. ***You may assume that the list pointed to by front is non-empty.***

Use the struct definition provided below.

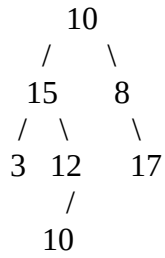
```
#include <stdlib.h>
#include <stdio.h>
#define MAX 1000

typedef struct node {
    int value;
    struct node* next;
} node;

int mode(node* front) {

    int freq[MAX], i;
    for (i=0; i<MAX; i++)
        freq[i] = 0;

    while (front != NULL) {
        freq[front->value]++;
        front = front->next;
    }

    int res = 0;
    for (i=1; i<MAX; i++)
        if (freq[i] > freq[res])
            res = i;

    return res;

}
```

Grading: 3 pts initializing frequency array, 4 pts filling frequency array, 3 pts finding index of maximum value of frequency array. Only take off 1 pt if ties are broken incorrectly.

Note: Please readjust points for solution ideas different than this as necessary.

4) (10 pts) DSN (Binary Trees)

For this problem you will write a modified inorder traversal of a binary tree. In a regular inorder traversal, you simply visit the nodes of the tree "in order". Consider the added task of adding up all of the numbers as you see them in the traversal, one by one, and printing out each intermediate sum. For example, if the input binary tree was:



the corresponding inorder traversal would visit 3, 15, 10, 12, 10(root), 8 and 17, in that order. For the added task, the traversal should print out 3, 18, 28, 40, 50, 58 and 75, respectively, the running sums after visiting each value.

Complete the function below, ***recursively***, so that is performs the given task. One way to accomplish this is to have the function take in a second value, `prevSum`, representing the previous sum of values prior to visiting the given node, and also to have the function return the sum of the nodes in its subtree.

```
typedef struct treenode {
    int value;
    struct treenode *left;
    struct treenode *right;
} treenode;

int inorderSum(treenode* root, int prevSum) {

    if (root == NULL) return 0;

    int sum = 0;
    sum += inorderSum(root->left, prevSum);

    sum += root->value;
    printf("%d ", sum+prevSum);

    sum += inorderSum(root->right, prevSum+sum);

    return sum;
}
```

Grading: There are 10 blanks, give 1 pt for each blank. If two blanks are slightly wrong, you may award 1 pt for both of them together.

5) (10 pts) ALG (Sorting)

For this question, implement the (very slow) sorting algorithm described below:

1. Randomly choose two distinct array indexes, i and j. (If i and j are equal, choose again.)
2. If array[min(i,j)] > array[max(i,j)], swap the two values.
3. Check if the array is sorted. If it's not, go back to step 1. If it is, return.

Recall that the function call `rand()` returns a random non-negative integer, so `rand()%n` will equal a random integer in between 0 and n-1.

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>

int min(int a, int b) {if (a < b) return a; return b;}
int max(int a, int b) {if (a > b) return a; return b;}

void randomSort(int* array, int length) {

    while (1) {

        int i = 0, j = 0;
        while (i == j) {
            i = rand()%length;
            j = rand()%length;
        }
        int minI = min(i,j);
        int maxI = max(i,j);

        if (array[minI] > array[maxI]) {
            int temp = array[minI];
            array[minI] = array[maxI];
            array[maxI] = temp;
        }

        int sorted = 1;
        for (i=0; i<length-1; i++)
            if (array[i] > array[i+1])
                sorted = 0;

        if (sorted) break;
    }

}
```

Grading: 3 pts picking random indices, 3 pts performing swap, if necessary, 3 pts check at end, 1 pt stopping/breaking if the array is sorted

Computer Science Foundation Exam

May 6, 2016

Section II A

DISCRETE STRUCTURES

NO books, notes, or calculators may be used,
and you must work entirely on your own.

SOLUTION

Question	Max Pts	Category	Passing	Score
1	15	PRF (Induction)	10	
2	10	PRF (Logic)	7	
3	15	PRF (Sets)	10	
4	10	NTH (Number Theory)	7	
ALL	50		34	

You must do all 4 problems in this section of the exam.

Problems will be graded based on the completeness of the solution steps and not graded based on the answer alone. Credit cannot be given unless all work is shown and is readable. Be complete, yet concise, and above all be neat.

1) (15 pts) PRF (Induction)

Use strong induction to prove that, for every non-negative integer, n, F_n is even if and only if 3 divides n. Here F_n is the Fibonacci sequence given by the recurrence:

$$F_0 = 0$$
$$F_1 = 1$$
$$F_n = F_{n-1} + F_{n-2}$$

Let $P(n)$ be the open statement F_n is even if and only if 3 divides n.

Basis Step:
Case $P(0)$:
$F_0 = 0$ and 0 is even. We also know $3 \mid 0$.

Case $P(1)$:
$F_1 = 1$ and 1 is odd. We also know $3 \nmid 1$.

Inductive Hypothesis:
Assume for some arbitrary positive integer k, that for all $0 \leq j \leq k$, $P(j)$ is true. That is:

F_j is even $\Leftrightarrow 3 \mid j$

Inductive Step: Show that $P(k+1)$ follows:

We must show two directions $3 \mid k+1 \Rightarrow F_{k+1}$ is even and F_{k+1} is even $\Rightarrow 3 \mid k+1$. The second case we will show through the contrapositive.

Case $(3 \mid k+1)$: In this case $3 \nmid k-1$ and $3 \nmid k$. From our inductive hypothesis F_{k-1} and F_k are both odd. The sum of two odd integers is even. Thus, F_{k+1} is even.

Case $(3 \nmid k+1)$: In this case either $3 \mid k$ or $3 \mid k-1$ but not both. From our inductive hypothesis either F_{k-1} or F_k must be odd and the other even. The sum of an odd integer and an even integer is odd. Thus, F_{k+1} is odd.

As we have shown $3 \mid k+1 \Rightarrow F_{k+1}$ is even and $3 \nmid k+1 \Rightarrow F_{k+1}$ is odd, it holds that $3 \mid k+1 \Leftrightarrow F_{k+1}$ is even or in other words $P(k+1)$ is true.

By the principle of mathematical induction, $P(n)$ is true for all non-negative integers, n.

Note: An alternative solution that breaks down F_{k+1} to a combination of F_{k-1} and F_{k-2} requires 3 base cases instead of 2.

Grading: Base case - 3 pts, IH - 2 pts, IS - 2 pts, Case (3 | (k+1) - 4 pts, Case $(3 \nmid k+1)$ - 4 pts.

2) (10 pts) PRF (Logic)

Here we have discovered some "rules of inference" that aren't valid. Invalidate them by finding counter-examples that make each premise true but make the conclusion false.

(a) $p \vee q$
$\neg p \vee \neg q$
$\therefore p$

Let $p = F$, $q = T$. This is the only counterexample that works. In this case, both premises are true (first is true due to q, second is true due to p), and the conclusion is false.

Grading: 4 pts, all or nothing, no explanation necessary.

(b) $(p \rightarrow q) \rightarrow r$
$\neg r$
$\therefore \neg p$

Let $p = T$, $q = F$, $r = F$. This is also the only counterexample possible. In this case, the first premise is true because the left hand side is false, which makes the whole premise true. The second premise is true since r is false. Finally, the conclusion is false since p is true.

Grading: 4 pts, all or nothing, no explanation necessary. Note - an alternative solution takes the two premises and uses them to prove that p must follow. This solution ALSO earns full credit.

(c) $p \rightarrow (q \rightarrow r)$
$\neg r$
$\therefore p$

Let $p = F$, $q = T/F$, $r = F$. Here p and r must be false but q can be either for a counterexample.

Grading: 2 pts, all or nothing, no explanation necessary.

3) (15 pts) PRF (Sets)

Show for finite sets A, B, C that if $B \cup C \subseteq A$ and $A \times B \subseteq A \times C$, then $B \subseteq C$.

We must show an arbitrary element of B must also be in C. Let $b \in B$ arbitrarily. This element b must also be an element of $B \cup C$. As $b \in B \cup C$ and $B \cup C \subseteq A$, it follows that $b \in A$.

From Cartesian product's definition, since $b \in A$ and $b \in B$, $(b, b) \in A \times B$. Now it follows from $A \times B \subseteq A \times C$ that $(b, b) \in A \times C$. This means that $b \in A$ and $b \in C$. Completing our proof since we've shown that $b \in C$.

QED

Grading: 3 pts for stating what needs to be proved and starting with a premise of an element in B. 3 pts for concluding that the element is in A. 2 pts for concluding that (b,b) is in A x B. 2 pts for finishing the proof.

Disprove for finite sets A, B, C that if $A \times B \subseteq A \times C$, then $B \subseteq C$.

Let $A = \emptyset$, $B = \{1, 2\}$, $C = \{2\}$. In this case, both $A \times B$ and $A \times C$ are the empty set since A is the empty set, making the premise true, but B isn't a subset of C since B contains 1 and C doesn't.

Grading: 4 pts counter-example, 1 pt explanation

4) (10 pts) NTH (Number Theory)

Prove that for any two distinct primes a and b there exists some prime c distinct from both a and b such that $c \mid (a + b)$.

There are two cases we must consider:

Case 1:
Suppose that a and b are both odd primes. As a and b are both odd, $a + b$ must be even. This means the prime number 2 divides $a + b$. As 2 equals neither a nor b we have found a c meeting our criteria.

Case 2:
There is only one even prime, 2. Let $a = 2$ without loss of generality and b be some other odd prime. We will show that $2 + b$ is divisible by some prime neither 2 nor b by contradiction.

Suppose 2 and b are the only prime divisors of $2 + b$. The number $2 + b$ is odd as 2 is even and b is odd. Thus 2 is not a divisor of $2 + b$. We also know that b does not divide $2 + b$ by the division algorithm as $2 + b = (1)b + 2$, $0 \leq 2 < b$. But this contradicts that $2 + b$ must be representable by the product of primes by the fundamental theorem of arithmetic. Therefore, some other prime p must divide $2 + b$. Alternatively, we can argue that $(2 + b) \equiv 2(mod b)$ because b is greater than 2, proving that b isn't a divisor of $(2 + b)$.

QED

Alternate proof: Since a ≠ b, it follows that a doesn't divide evenly into a+b and b doesn't divide evenly into a+b, since a ∤ b and b ∤ a. (A slightly more formal proof is showing that a+b≡a (mod b) and since a and b are distinct primes, a ≢ 0 (mod b), and vice versa.)

By the Fundamental Theorem of Arithmetic, if follows that a+b has a unique prime factorization, and must have at least one prime factor. Since that prime factor isn't a or b, a distinct prime factor c must exist.

Grading: Notes - there are other ways to prove these. Criteria for this solution:

5 pts even case, 5 pts odd case

Alternate criteria solution - 5 pts explanation that a and b don't divide a+b, 5 pts for concluding that a different prime must.

Computer Science Foundation Exam

May 6, 2016

Section II B

DISCRETE STRUCTURES

NO books, notes, or calculators may be used,
and you must work entirely on your own.

SOLUTION

Question	Max Pts	Category	Passing	Score
1	15	CTG (Counting)	10	
2	10	PRB (Probability)	7	
3	15	PRF (Functions)	10	
4	10	PRF (Relations)	7	
ALL	50		34	

You must do all 4 problems in this section of the exam.

Problems will be graded based on the completeness of the solution steps and not graded based on the answer alone. Credit cannot be given unless all work is shown and is readable. Be complete, yet concise, and above all be neat.

1) (15 pts) CTG (Counting)

Please leave your answers in factorials, permutations, combinations and powers. Do not calculate out the actual numerical value for any of the questions. Justify your answers.

(a) (5 pts) If seven people are to be seated in a single row with ten chairs, how many unique arrangements are possible? (For example, if the people are labeled P_1 through P_7 and we use B to denote an empty chair, then one possible arrangement is P_1, B, B, P_2, P_4, P_3, P_7, P_5, B, P_6. Notice that the people are distinguishable but the empty chairs aren't.)

One process for counting the possibilities (where our guests are "A" through "G") is:

- Seat Guest A – 10 choices
- Seat Guest B – 9 choices
- Seat Guest C – 8 choices
- Seat Guest D – 7 choices
- Seat Guest E – 6 choices
- Seat Guest F – 5 choices
- Seat Guest G – 4 choices

Naturally, all the remaining seats are left empty. By the rule of product, the number of unique arrangements is $\frac{10!}{3!}$. **(Grading - 2 pt product, 3 pts terms)**

(b) (10 pts) How many of the arrangements from part (a) guarantee that none of the empty chairs are adjacent to one another?

Let x_1 equal the number of chairs left of the first empty chair, x_2 equal the number of chairs in between the first two empty chairs, x_3 be the number of empty chairs in between the last two empty chairs and x_4 equal the number of chairs to the right of the last empty chair. The number of non-negative integer solutions to the equation below is the number of ways to choose the empty chairs:

$$x_1 + x_2 + x_3 + x_4 = 7$$

such that $x_2 > 0$ and $x_3 > 0$. We can deal with these restrictions by simply setting up new non-negative integer variables $x_2 = x_2' + 1$ and $x_3 = x_3' + 1$, where $x_2' \geq 0$ and $x_3' \geq 0$. Thus, now we want to find the number of non-negative integer solutions to:

$$x_1 + x_2' + 1 + x_3' + 1 + x_4 = 7$$
$$x_1 + x_2' + x_3' + x_4 = 5$$

Using the combinations with repetition formula, we find that the number of non-negative solutions to the equation above is $\binom{5+4-1}{4-1} = \binom{8}{3}$.

Finally, the 7 people can be placed in the 7 non-empty chairs in 7! ways, so the final answer is $\binom{8}{3}7!$.

Grading: 7 pts for # of ways to choose empty chairs (many ways to do this, give partial as you see fit), 2 pts for seating the people, 1 pt for multiplying the two.

2) (10 pts) PRB (Probability)

A mad scientist has created a probability-driven lock that can be permanently affixed to all manner of things (gym bags, lockers, vehicles, door locks, and so on). The lock has a five-digit LED display and a single button, and works as follows:

Once the lock is closed, if you want to open it, you press the button, and it generates a random sequence of five digits on its display. (Each digit is a random integer on the range zero through nine.) If the product of all five digits is odd, the lock opens. Otherwise, you have to press the button again to generate another random sequence of digits.

If you press the button three times without getting the lock to open, it seals itself shut forever.

If you put one of these locks on something, what is the probability that you'll actually be able to get it to open again?

The key insight here is that if any one of the five digits is even, then you have a factor of two in your overall product, and therefore the product is even. In order for the product to be odd, all five digits must be odd.

Since each of the five digits can either be even or odd (each with equal probability), we can consider the size of the sample space for a single press of the lock's button to be $2^5 = 32$. There is only one way for the product to be odd (that is, for all the digits to be even), so the probability of an odd product is $\frac{1}{32}$ = 0.03125. This means the probability of an even product is $\frac{31}{32}$.

Another way to conceive of the sample space is to look at all possible five-digit numbers that could pop up. There are $10^5 = 100{,}000$ such numbers. The number of outcomes where all five digits are odd is $5^5 = 3{,}125$, since there are five odd digits in the range zero through nine. So, the probability of an odd product is $\frac{3{,}125}{100{,}000} = 0.03125$.

The probability of getting three even products in a row is $\left(\frac{31}{32}\right)^3$. That also corresponds to the probability that the lock will be sealed shut forever. So, the probability that you'll actually succeed in opening the lock is $1 - \left(\frac{31}{32}\right)^3 \approx 9.085\%$.

Alternate Solution: Another way to arrive at that probability is to realize that you could open the lock on the first, second, or third try. These mutually disjoint events give rise to the following sum of probabilities:

$$\left(\frac{1}{32}\right) + \left(\frac{31}{32}\right)\left(\frac{1}{32}\right) + \left(\frac{31}{32}\right)\left(\frac{31}{32}\right)\left(\frac{1}{32}\right) \approx 9.085\%$$

Grading:
3 pts for realizing that the product is odd only if all five digits are odd.
3 pts for the cardinality of the sample space.
4 pts for their approach to the final answer (event complement or sum of probabilities)

Please award partial credit for each piece as appropriate.

3) (15 pts) PRF (Functions)

(a) (4 pts) Define finite sets A and B that satisfy ***all three*** of the following criteria simultaneously:

1. A and B are non-empty.
2. There exists a surjective function from A to B.
3. Every possible surjective function from A to B is also injective.

For this question, any sets A and B that are both non-empty **(2 pts)** and have the same cardinality **(2 pts)** will work.

(b) (4 pts) Define finite sets A and B that satisfy ***all three*** of the following criteria simultaneously:

1. A and B are non-empty.
2. There exists an injective function from A to B.
3. It is **impossible** to define an injective function from A to B that is also surjective.

For this question, any sets A and B that are both non-empty **(2 pts)** and where $|A| < |B|$ **(2 pts)** will work.

(c) (4 pts) Define finite sets A and B that satisfy ***all three*** of the following criteria simultaneously:

1. A and B are non-empty.
2. There exists an injective function from A to B.
3. It is **possible** to define an injective function from A to B that is also surjective.

For this question, any sets A and B that are both non-empty **(2 pts)** and where $|A| = |B|$ **(2 pts)** will work.

(d) (3 pts) Define finite sets A and B that satisfy ***both*** of the following criteria simultaneously:

1. A and B are non-empty.
2. It is **impossible** to define a function from A to B that is **not** injective.

For this question, A must contain a **single** element **(2 pts)**, and B must be non-empty **(1 pt)**.

4) (10 pts) PRF (Relations)

(a) (3 pts) What three properties must a relation satisfy in order to be a partial ordering relation?

It must be reflexive, transitive, and antisymmetric.

Grading:
-1 pt for each incorrect property listed
-1 pt for each correct property missing from the list

(b) (7 pts) Consider the relation $\mathcal{R}$ on $\mathbb{Z}^+$ defined as follows: For all positive integers x and y, $(x, y) \in \mathcal{R}$ if y is divisible by x. Prove or disprove that $\mathcal{R}$ is a partial ordering relation.

$\mathcal{R}$ is reflexive:
$x \mid x$ for all positive integers x, so $(x, x) \in \mathcal{R}$. Therefore, $\mathcal{R}$ is reflexive.

$\mathcal{R}$ is transitive:
If $(x, y) \in \mathcal{R}$ and $(y, z) \in \mathcal{R}$, then we know $x \mid y$ and $y \mid z$. It follows that $y = xm$ for some integer m, and $z = yn$ for some integer n. Thus, $z = (xm)n = x(mn)$, and since mn is an integer, we have that $x \mid z$. Therefore, $(x, z) \in \mathcal{R}$.

$\mathcal{R}$ is antisymmetric:
If $(x, y) \in \mathcal{R}$ and $(y, x) \in \mathcal{R}$, we know $x \mid y$ and $y \mid x$. Due to the former, there exists a positive integer c such that y = cx. Due to the latter, there exists a positive integer integer d such that x = dy. Substituting the second equation into the first, we find that y = cx = c(dy), so y = cdy. Since y must be non-zero, we get cd = 1. Since c and d are positive, it follows that both must be 1 and x and y are equal. Therefore, $\mathcal{R}$ is antisymmetric.
.

Grading:
1 pts for reflexive
3 pts for transitive
3 pts for antisymmetric

If they try to prove (or disprove) the wrong properties, try to award at least half credit for the properties they worked with.