Computer Science Foundation Exam

December 18, 2015

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			

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 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) (10 pts) DSN (Recursive Functions)

Write a ***recursive*** function that takes the root of a ternary tree (a tree where each node has at most three children) and determines whether all the nodes that have a middle child also have both a left child and a right child. If so, return 1. Otherwise, return 0. Note: If the function with a null input, the output should be 1.

The node struct and functional prototype for this question are:

```
typedef struct node
{
   char *data;
   struct node *left, *middle, *right;
} node;

int hasProperty(node *root)
{
  if (root == NULL)
     return 1;

  else if (root->middle == NULL)
     return hasProperty(root->left) && hasProperty(root->right);

  else if (root->left == NULL || root->right == NULL)
     return 0;

  else
     return hasProperty(root->left) &&
            hasProperty(root->right) &&
            hasProperty(root->middle);
}
```

Grading:

2 pts for root == NULL base case,
3 pts for handling root->middle == NULL case,
2 pts for handling case where exactly middle isn't NULL but left or right is
3 pts for handing case where none are NULL

2) (10 pts) ANL (Summations and Algorithm Analysis)

Give the big-oh runtimes for each of the following functions in terms of *n* and/or *k* (where *k* is the length of string *s*), given that *strlen*(*s*) is an O(*k*) function and *toupper*(*c*) is an O(1) function. You may assume that *s* is non-NULL and contains at least one character (so, it shouldn't cause any of the following functions to crash).

```
void uppercase(char *s)
{
   int i;
   for (i = 0; i < strlen(s); i++)
      s[i] = toupper(i);
}
```

uppercase run time: $\boldsymbol{O(k^2)}$

```
void uppercase_remix(char *s)
{
   int i, length = strlen(s);
   for (i = 0; i < length; i++)
      s[i] = toupper(i);
}
```

uppercase_remix run time: $\boldsymbol{O(k)}$

```
void uppercase_unreliable(char *s){
   int i, j = strlen(s) - 1, m;
   while (i <= j)
   {
      m = i + (j - i) / 2;
      if (rand() % 2 == 0)
      {
         s[i] = toupper(s[i]);
         i = m + 1;
      }
      else
      {
         s[j] = toupper(s[j]);
         j = m - 1;
      }
   }
}
```

uppercase_unreliable run time: $\boldsymbol{O(k) + O(\log k) = O(k)}$

```
void mad_scramble(char *s, int n){
   int i;
   for (i = 0; i < n; i++)
      s[strlen(s) - 1] = rand() % 25 + 'a';
}
```

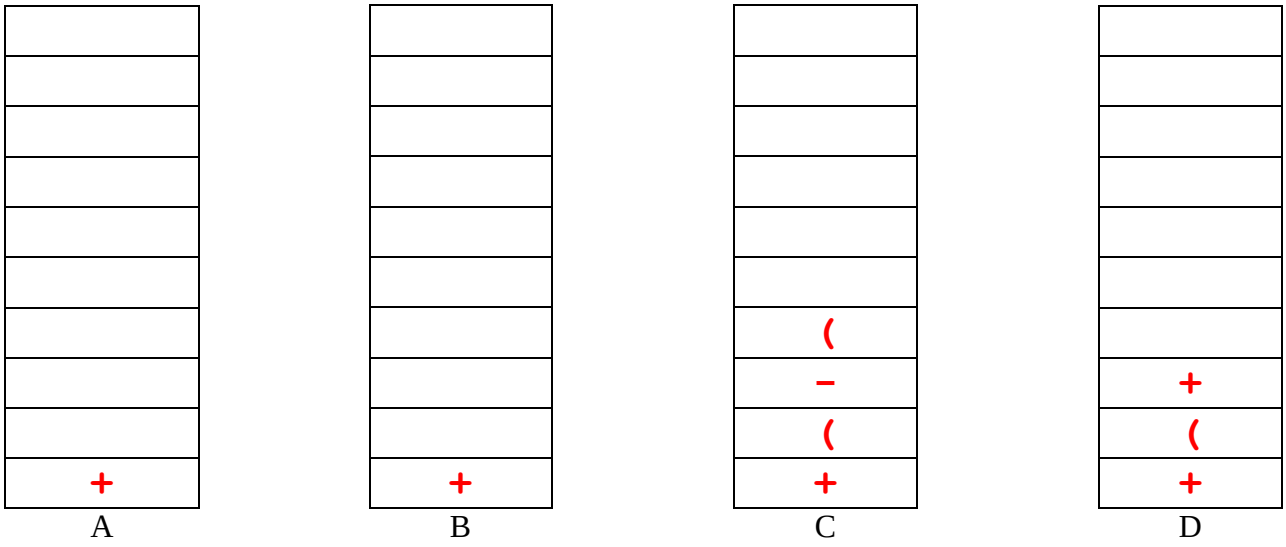
mad_scramble run time: $\boldsymbol{O(nk)}$

Grading: 3 pts uppercase, 2 pts uppercase_remix, 3 pts uppercase_unreliable, 2 pts mad_scramble, each part is all or nothing.

3) (10 pts) ALG (Stacks)

Use a stack to convert the following infix expression to a postfix expression. Please show the state of the stack at the exact point in time when the algorithm reaches the marked locations (A, B, C, and D) while processing the expression.

```
                A           B           C               D
8  *  3  /  2  +  9  -  6  +  (  3  -  (  8  *  2  )  +  7  )
```



Equivalent Postfix Expression:

```
8  3  *  2  /  9  +  6  -  3  8  2  *  -  7  +  +
```

Grading: 1 pt for each stack, 6 pts for the expression (take off 1 pt per error in the expression, capping at 6)

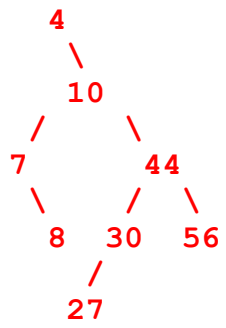
4) (10 pts) ALG (Binary Search Trees and Hash Tables)

a) (8 pts) Draw a **single** binary search tree that gives rise to all three of the following tree traversals:

Inorder: 4 7 8 10 27 30 44 56

Preorder: 4 10 7 8 44 30 27 56

Postorder: 8 7 27 30 56 44 10 4



Grading: 8 pts total, 1 pt per node placement, try to adjust for errors so that you don't take off multiple points for one incorrect node; namely, grade relative to an initial error they made.

b) (2 pts) If we insert an element into a hash table using quadratic probing to resolve collisions, what two conditions must be met to ensure that if an open spot exists in our hash table, we will find that spot (rather than getting stuck in an infinite loop)?

1. **The table size must be a prime number.**
2. **The table must be at least half empty.**

Grading: 1 pt each

5) (10 pts) ALG (Base Conversion)

Write a function that takes a string *str* and an integer *b* (where $2 \leq b \leq 10$), and returns 1 if *str* represents an integer in base *b* that is a perfect power of *b*. For example:

```
isPower("323", 4);      // Return 0. 323₄ = 59₁₀, which is not a power of 4
isPower("27", 3);       // Return 0. 27 is not a valid base 3 integer.
isPower("plum", 8);     // Return 0. plum is not a valid base 8 integer.
isPower("1000", 10);    // Return 1. 1000₁₀ is a power of 10 (10³)
isPower("000001", 2);   // Return 1. 1₂ = 1₁₀, which is a power of 2 (2⁰)
```

Notes: You may assume *b* is always within the range specified above. Your function must return 0 if *str* is NULL or the empty string. Strings may be padded on the left with any number of zeros.

```
// You must use this function signature. You may write helper functions as
// needed:
int isPower(char *str, int b);

int charToInt(char c) { return c - '0'; }

int isPower(char *str, int b)
{
   int i, length, c, flag = 0;

   if (str == NULL || str[0] == '\0')
      return 0;

   for (i = 0; i < strlen(str); i++)
   {
      // Convert this character to an integer.
      c = charToInt(str[i]);

      // If this is not even a valid digit in base b, return 0.
      if (c > b - 1 || c < 0)
         return 0;

      // Key insight: For this to be a perfect power of b, we must
      // have '1' followed by '0''s only.
      if (c == 1)
      {
         if (flag > 0)  // If we have seen more than one '1', it's over.
            return 0;
         flag = 1;
      }
      // Can't have anything but 0's and 1's.
      else if (c != 0)
      {
         return 0;
      }
   }
   return flag;
}
```

Grading: 4 pts - for taking care of invalid cases, 1 pt - returns 1 or 0 for all cases, 2 pts - rejects strings that have anything but 0 or 1, 2 pts - rejects any string with > 1 non-zero char, 1 pt - accepts correct strings

Computer Science Foundation Exam

December 18, 2015

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)

Consider the recursive function `sum` shown below:

```
double sum(int* array, int low, int high){
    if (low == high)
        return array[low];
    int mid = (low+high)/2, left = 0, right = 0, i;
    for (i=low; i<=mid; i++) left += array[i];
    for (i=mid+1; i<=high; i++) right += array[i];
    if (left > right) return left + sum(array, low, mid);
    return right + sum(array, mid+1, high);
}
```

(a) (3 pts) Let T(n) represent the run time of the function call `sum(array, 0, n-1)`, where array is an integer array of size n. Write a recurrence relation that T(n) satisfies.

$$T(n) = T\left(\frac{n}{2}\right) + O(n), T(1) = 1$$

Grading: 2 pts T(n/2), 1 pt O(n).

(b) (7 pts) Using the iteration method, determine a closed-form solution (Big-Oh bound) for T(n). Assume T(1) = O(1).

$$T(n) = T\left(\frac{n}{2}\right) + cn$$
$$T(n) = T\left(\frac{n}{4}\right) + \frac{cn}{2} + cn$$
$$T(n) = T\left(\frac{n}{4}\right) + \frac{3cn}{2}$$
$$T(n) = T\left(\frac{n}{8}\right) + \frac{cn}{4} + \frac{3cn}{2}$$
$$T(n) = T\left(\frac{n}{8}\right) + \frac{7cn}{4}$$

After k steps, we have

$$T(n) = T\left(\frac{n}{2^k}\right) + \frac{(2^k - 1)cn}{2^{k-1}}$$

Plugging in $\frac{n}{2^k} = 1$, we have

$$T(n) = T(1) + \frac{(n-1)cn}{\frac{n}{2}}$$

$$T(n) = 1 + 2c(n-1)$$

$$T(n) = O(n)$$

Grading: 1 pt each iteration (3 pts total), 2 pts general form with k, 2 pts finishing problem. Note in place of *cn* students can write *O(n)* or even *n*, with no penalty.

2) (10 pts) ANL (Algorithm Analysis)

(a) (5 pts) A matrix factorization algorithm that is run on a input matrix of size *n* x *n*, runs in O(n^3) time. If the algorithm takes 54 seconds to run for an input of size 3000 x 3000, how long will it take to run on an input of size 1000 x 1000?

Let T(n) be the run-time of the algorithm on a matrix input of size n x n. We have:

$$T(3000) = c3000^3 = 54\ sec$$
$$c = \frac{54}{27 \times 10^9} sec = \frac{2}{10^9} sec$$

We desire to find T(1000):

$$T(1000) = c(1000^3) = \frac{2\ sec}{10^9} \times 10^9 = 2\ sec$$

Grading: 2 pts solving for c, 2 pts plugging in c, 1 pt for simplifying to 2 sec. Ratio method is valid as well, map points accordingly.

(b) (5 pts) A string algorithm with inputs of lengths *n* and *m* runs in O(n^2m) time. If the algorithm takes 2 seconds to run on an input with *n* = 1000 and *m* = 500, how long will the algorithm take to execute on an input with *n* = 250 and *m* = 1000?

Let T(n, m) be the run-time of the algorithm on strings inputs with lengths n and m. We have:

$$T(1000, 500) = c(1000^2)(500) = 2\ sec$$
$$c = \frac{2\ sec}{5 \times 10^8}$$

We desire to find T(250, 1000):

$$T(250, 1000) = c(250^2)(1000) = \frac{2\ sec}{5 \times 10^8} \times 625 \times 10^5 = \frac{250}{1000} sec = .25\ sec$$

Grading: 2 pts solving for c, 3 pts obtaining final answer. Ratio method is valid as well, map points accordingly.

3) (10 pts) DSN (Linked Lists)

Write a **recursive** function, `aboveThreshold`, that takes in a pointer to the front of a linked list storing integers, and an integer, `limit`, and returns the number of values stored in the linked list that are strictly greater than `limit`. For example, if the function was called on a list storing 3, 8, 8, 6, 7, 5, 7, 9 and `limit` equaled 6, then the function should return 5, since the 2nd, 3rd, 5th, 7th and 8th values in the list are strictly greater than 6. (Notice that we don't count the 4th element.)

Use the struct definition provided below.

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

int aboveThreshold(node* front, int limit) {

    if (front == NULL) return 0;

    int cnt = 0;
    if (front->data > limit) cnt = 1;

    return cnt + aboveThreshold(front->next, limit);
}
```

Grading conceptually: 3 pts for the base case, 3 pts for adding 1 in the case that the first item is greater than the limit, 3 pts for adding in the appropriate recursive call, 1 pt for returning. Max of 3 pts for an iterative solution.

4) (10 pts) DSN (Binary Trees)

We define the ***offcenter value*** for each node in a binary tree as being the absolute value of the difference between the height of its left subtree and the height of its right subtree. For example, for an AVL tree, each node has an offcenter value of 0 or 1. Also, note that we define the offcenter value of a null node to be 0. Write a function, `maxOffCenterValue` that computes the maximum offcenter value of any node in a tree pointed to by root. To make your task easier, assume that the height of each node is stored in the corresponding struct for that node in the component `height`.

Using the struct definition given below, complete the function in the space provided.

```
#include <math.h>

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

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

int maxOffCenterValue(treenode* root) {

    if (root == NULL) return 0;
    if (root->left == NULL && root->right == NULL) return 0;
    if (root->left == NULL || root->right == NULL)
        return root->height;

    int lVal = maxOffCenterValue(root->left);
    int rVal = maxOffCenterValue(root->right);
    int res = max(lVal, rVal);

    int cur = abs(root->left->height - root->right->height);
    return max(res, cur);
}
```

Grading: 1 pt NULL case, 1 pt 1 node case, 2 pts root has one child, 1 pt for left rec call, 1 pt for right rec call, 2 pts for current node calculation, 2 pts for getting max of all three.

5) (10 pts) ALG (Sorting)

Write the code for any one of the following $O(n^2)$ sorts: Bubble Sort, Insertion Sort, Selection Sort in a single function below. Your code should sort the array from smallest to largest. (Namely, after your code finishes array[i] ≤ array[i+1] for all i, 0 ≤ i < length-1.) Please provide the name of the sort you are choosing to implement and fill in the function prototype below.

```
void bubblesort(int* array, int length) {
    int i,j;
    for (i=length-1; i>0; i--) {
        for (j=0; j<i; j++) {
            if (array[j] > array[j+1]) {
                int temp = array[j];
                array[j] = array[j+1];
                array[j+1] = temp;
            }
        }
    }
}

void insertionsort(int* array, int length) {
    int i,j;
    for (i=1; i<length; i++) {
        j = i;
        while (j>0 && array[j] < array[j-1]) {
            int temp = array[j];
            array[j] = array[j-1];
            array[j-1] = temp;
            j--;
        }
    }
}

void selectionsort(int* array, int length) {
    int i,j;
    for (i=length-1; i>=0; i--) {
        int bestJ = 0;
        for (j=1; j<=i; j++) {
            if (array[j] > array[bestJ])
                bestJ = j;
        }
        int temp = array[i];
        array[i] = array[bestJ];
        array[bestJ] = temp;
    }
}
```

Grading: 3 pts outer loop structure, 4 pts inner loop structure, 3 pts appropriate swapping, 1 pt off for sorting properly but using the wrong name for the sort or writing an extra function.

Computer Science Foundation Exam

December 18, 2015

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	15	PRF (Logic)	10	
3	10	PRF (Sets)	6	
4	10	NTH (Number Theory)	6	
ALL	50		32	

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 mathematical induction to prove that, for every positive integer n,

$$5 \mid (8^n - 3^n)$$

Base Case (n = 1):
$8^1 - 3^1 = 5$, which is divisible by 5, so the base case holds. **(Grading: 2 pts)**

Inductive Hypothesis:
Assume for some arbitrary positive integer k, that $5 \mid (8^k - 3^k)$. That is to say, there exists some integer p such that $8^k - 3^k = 5p$. **(Grading: 2 pts)**

Inductive Step:
We want to show that $5 \mid (8^{k+1} - 3^{k+1})$. In other words, show that $8^{k+1} - 3^{k+1}$ can be written as $5q$, for some integer q. **(Grading: 2 pts)**

$8^{k+1} - 3^{k+1}$	$= (8)8^k - (3)3^k$	**(Grading: 2 pts)**
	$= (5+3)8^k - (3)3^k$	**(Grading: 2 pts)**
	$= (5)8^k + (3)8^k - (3)3^k$	**(Grading: 1 pts)**
	$= (5)8^k + (3)(8^k - 3^k)$	**(Grading: 1 pts)**
	$= (5)8^k + (3)(5p)$	by the inductive hypothesis **(Grading: 2 pts)**
	$= (5)(8^k + 3p)$	**(Grading: 1 pt)**
	$= (5)(q)$	where $q = 8^k + 3p$, which is an integer, since the set of integers is closed under multiplication and addition ($8^k \in \mathbb{Z}$ as $k \in \mathbb{Z}^+$)

So, $5 \mid (8^{k+1} - 3^{k+1})$.

Thus, by the principle of mathematical induction, $5 \mid (8^n - 3^n)$ for all positive integers, n. ■

2) (15 pts) PRF (Logic)

Validate the following argument using the laws of logic, substitution rules or rules of inference. List the rule used in each step and label the steps used in each derivation.

$$\begin{array}{l} (\neg q \vee n) \rightarrow r \\ q \\ z \rightarrow y \\ p \\ t \rightarrow \neg q \\ \hline \therefore w \rightarrow ((p \vee \neg r) \wedge \neg t) \end{array}$$

Proof:

1.	$t \rightarrow \neg q$	Premise
2.	$q \rightarrow \neg t$	Contraposition (and Double Negation) on (2)
3.	q	Premise
4.	$\neg t$	Modus Ponens on (2) and (3)
5.	p	Premise
6.	$p \vee \neg r$	Disjunctive Amplification on (5)
7.	$(p \vee \neg r) \wedge \neg t$	Rule of Conjunction on (6) and (4)
8.	$\neg w \vee ((p \vee \neg r) \wedge \neg t)$	Disjunctive Amplification on (7)
9.	$w \rightarrow ((p \vee \neg r) \wedge \neg t)$	Definition of Implication (and Double Negation) on (9)

Grading: 5 pts for deriving not t, 3 pts for deriving p or not r, 3 pts for deriving (p or not r) and not t, 4 pts for deriving the final conclusion, take off a maximum of 4 pts for not listing reasons, taking off 1 pt per missing reason or incorrect reason stated

3) (10 pts) PRF (Sets)

a) Let $C = \{x, y\}$. What is $C \times C$?

$C \times C = \{(x,x), (x,y), (y,x), (y,y)\}$

b) Let $D = \{p, o, m\}$. What is $\mathcal{P}(D)$ (the power set of D)?

$\mathcal{P}(D) = \{\emptyset, \{p\}, \{o\}, \{m\}, \{p,o\}, \{p,m\}, \{o,m\}, \{p,o,m\}\}$

c) Let A and B be finite sets. Then, give the following in terms of $|A|$ and $|B|$:

$|A \times B| = |A| \cdot |B|$

$|\mathcal{P}(A)| = 2^{|A|}$

$|\mathcal{P}(A \times B)| = 2^{|A| \cdot |B|}$

d) Let A and B be finite sets. Give the following in terms of $|A|$, $|B|$, $|A \times B|$, and/or $|A \cap B|$. (Do not use $|A \cup B|$ in your answer.)

$|\mathcal{P}(A \cup B)| = 2^{|A|+|B|-|A \cap B|}$

a) Under what conditions does $A \times B$ equal to $B \times A$? (To receive full credit, your answer must cover all possible conditions where this occurs.)

Either A = B, or exactly one of A or B is the empty set. (The cross product of the empty set with any non-empty set is simply the empty set.)

(If both A and B are the empty set, then that falls under A = B.)

Grading for each part: 2 pts if completely correct, 1 pt if portions of the answer are correct, 0 pts otherwise, total is 10 pts for 5 parts

4) (10 pts) NTH (Number Theory)

Show that for all integers a and b, if $30 \mid (14a + 8b)$, then $30 \mid (2a - 46b)$.

There are a few ways to solve this one, but one is to observe that $2a$ differs from $30a$ by $28a$. From there, we have:

$30 \mid (14a + 8b)$, so $30 \mid 2(14a + 8b) = 28a + 16b$.

Clearly, $30 \mid 30a$. Since $30 \mid 30a$ and $30 \mid (28a + 16b)$, $30 \mid (30a - (28a + 16b)) = 2a - 16b$.

Also, $30 \mid 30b$. Since $30 \mid 30b$ and $30 \mid (2a - 16b)$, $30 \mid (2a - 16b - 30b) = 2a - 46b$. ■

Grading: 3 pts for trying some multiple of 14a + 8b, 2 pts for trying to add or subtract multiples of 30a, 2 pts for trying to add or subtract multiples of 30b, 3 pts for finishing up the problem.

Computer Science Foundation Exam

December 18, 2015

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	15	PRB (Probability)	10	
3	10	PRF (Functions)	6	
4	10	PRF (Relations)	6	
ALL	50		32	

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.

Consider a string s of lowercase latin letters of length 10.

(a) (5 pts) Suppose we require s to have no two consecutive letters that are the same. How many such strings s exist?

There are 26 ways to place the first character of s. Each choice after that must be different than the previous character, so 25^9 ways to place remaining characters. Using rule of product, we derive $26 \cdot 25^9$ total ways.

Grading: 2 pts first char, 2 pts other 9 chars, 1 pt mult terms

(b) (5 pts) Suppose we require s to have each consecutive triplet of letters be pairwise distinct. How many such strings s exist?

There are 26 ways to place the first character. We must differ from the first character with the second character, so this choice has 25 ways. The remaining 8 character must differ from the previous two characters so there are 24^8 ways to choose remaining characters. By rule of product, we derive $26 \cdot 25 \cdot 24^8$ ways.

Grading: 1 pt first char, 1 pt second char, 2 pts other 9 chars, 1 pt mult terms

(c) (5 pts) Suppose we require s to have no two consecutive letters that are the same but additionally require that s be a palindrome. (A string that reads the same forward or backwards.) How many such strings s exist?

No two consecutive characters can be equal, but as s is an even length palindrome the middle two characters must be equal. This we have a contradiction in our requirements, resulting in 0 ways to form palindromes with the consecutive character requirement.

Grading: full credit for correct answer, 2 pts for answers that count # of arrangements of the first five letters, give other partial credit as you deem necessary

2) (15 pts) PRB (Probability)

Suppose we roll a 4 sided die with the numbers [1,4] written on them. After the first die roll we roll the die k times where k is the number on the first die roll. The number of points you score is the sum of the face-values on all die rolls (including the first). What is the expected number of points you will score?

Let $E_r(k)$ be the expected score if you roll k dice. Suppose we have k dice. Let X_i be the discrete random variable representing the i^{th} die's score. By linearity of expectations $E(X_1 + X_2 + X_3 + X_4) = E(X_1) + E(X_2) + E(X_3) + E(X_4)$. As the die doesn't change we can calculate $E(X_i)$ using the standard formula for expected value: $E(X_i) = 0.25(1) + 0.25(2) + 0.25(3) + 0.25(4) = \frac{10}{4}$. Now we can calculate $E_r(k) = k \cdot \frac{10}{4}$. **(Grading: 5 pts, 1 per term and 1 for adding properly)**

Let X be a discrete random variable representing the expected score total. We can calculate $E(X)$ be calculating each expected value after knowing the outcome of the first die. This value is

$$E(X) = 0.25\big(1 + E_r(1)\big) + 0.25\big(2 + E_r(2)\big) + 0.25\big(3 + E_r(3)\big) + 0.25\big(4 + E_r(4)\big)$$
$$= 0.25\left(10 + \frac{10}{4}(1 + 2 + 3 + 4)\right) = 0.25\left(10\left(1 + \frac{10}{4}\right)\right) = \frac{35}{4} = 8.75$$

There also exists a solution using probability trees. **(Grading: 10 pts, 2 pts for each case and 2 pts for adding everything up properly)**

Grading Note: Other solutions may exist, please try to award partial credit for significantly different approaches as you see fit.

3) (10 pts) PRF (Functions)

The function $C(n,k)$ is known as the choose function.
Let $X = \{(n,k) | n \in \mathbb{N} \wedge k \in \mathbb{N} \wedge k \leq n\}$. (Note that $\mathbb{N}$ is the non-negative integers.)
For the purposes of this problem we define the choose function, C, as follows:

$$C: X \to \mathbb{Z}^+, C(n,k) = \frac{n!}{k!\,(n-k)!}$$

(a) (5 pts) Is C injective? Prove or disprove this property.

No! We must show that $f(a) = f(b) \nRightarrow a = b$. Let $a = (3,2)$ and $b = (3,1)$. It is clear that $C(3,2) = C(3,1)$:

$$C(3,2) = \frac{3!}{1!\,2!} = C(3,1)$$

So $f(a) = f(b)$. Yet $(3,2) \neq (3,1)$.

Grading: 0 pts for answering yes, 2 pts for answering no, 3 pts for a specific example where f(a,b) = f(c, d) but either a ≠ c or b ≠ d.

(b) (5 pts) Is C surjective? Prove or disprove this property.

Yes. To do this we must show that there exists (n,k) such that $C(n,k) = z, z \in \mathbb{Z}^+$ for all $z \in \mathbb{Z}^+$.

Let $z \in \mathbb{Z}^+$ arbitrarily. As $\mathbb{Z}^+ \subseteq \mathbb{N}$, find $C(z,1) = \frac{z!}{(z-1)!1!} = \frac{z}{1} = z$.
As we have shown $C(z,1) = z$, we have found $(n,k) \in \mathbb{N}^2$ that hits each element of $\mathbb{Z}^+$.

QED

Grading: 0 pts for answering no, 2 pts for answering yes, 3 pts for showing input values and b such that for an arbitrary positive integer y, f(a, b) = y. You may give partial credit on this part as well.

4) (10 pts) PRF (Relations)

Consider the relation $\mathcal{R}$ on $\mathbb{Z}^2$ where $\big((a,b),(c,d)\big) \in \mathcal{R}$ when $ad - bc \geq 0$. Determine (with proof) if $\mathcal{R}$ meets or doesn't meet each of these properties: reflexive, symmetric, anti-symmetric, transitive.

$\mathcal{R}$ is reflexive: Show $\big((a,b),(a,b)\big) \in \mathcal{R}$. $ab - ba = 0 \geq 0$. Thus $\mathcal{R}$ is reflexive.

$\mathcal{R}$ is not symmetric: $\big((3,1),(1,3)\big) \in \mathcal{R}$ as $3 \cdot 3 - 1 \cdot 1 = 9 - 1 = 8 \geq 0$, yet
$\big((1,3),(3,1)\big) \notin \mathcal{R}$ as $1 \cdot 1 - 3 \cdot 3 = 1 - 9 = -9 < 0$.

$\mathcal{R}$ is not anti-symmetric: $\big((1,0),(-1,0)\big) \in \mathcal{R}$ as $1(0) - 0(-1) = 0 - 0 = 0 \geq 0$.
$\big((-1,0),(1,0)\big) \in \mathcal{R}$ as $-1(0) - 0(1) = 0 - 0 = 0 \geq 0$.
Yet, $(1,0) \neq (-1,0)$.

$\mathcal{R}$ is not transitive:

$\big((1,0),(0,1)\big) \in \mathcal{R}$ as $1 \cdot 1 - 0 \cdot 0 = 1 - 0 = 1 \geq 0$.
$\big((0,1),(0,-1)\big) \in \mathcal{R}$ as $0(-1) - 1(0) = 0 - 0 \geq 0$.

Yet $\big((1,0),(0,-1)\big) \notin \mathcal{R}$ as $1(-1) - 0 \cdot 0 = -1 - 0 = -1 < 0$.

In other words, sorting by the cross product is a bad idea.

Grading: 1 pt reflexive, 3 pts for each of the others, for the ones that are 3 pts 1 pt for saying no, 2 pts for finding a counter-example