CpSc 421 Final **150 minutes**

Do any **eight** of the **ten** problems below. If you attempt more than eight problems, please indicate which ones to grade (otherwise we will make a random choice). This allows you to attempt 104 points. The grading will be on a scale of 100. Thus, there are 4 points of extra credit built into the exam.

Note that each problem is worth 13 points. While the difficulty of a problem is a subjective quantity, I expect that most of you will find that the difficulty of the problems vary even though they all receive the same credit. I have ordered the problems by the length of the question – I don't expect that you'll find any significant correlation between this and the difficulty (or length, for that matter) of the solution. My solutions range from two to thirteen sentences in length; although this range may change slightly by the time that I post their final version. My solutions tend to be longer than necessary; so, you should take this as an upper bound on what is needed. I will take off points for answers that run-on with irrelevant stuff hoping to pick up partial marks – your solutions should have clear connections to the problems.

**Bug bounties** are in effect. If you find an error that affects the solution of the exam, raise your hand and bring it to my attention. Please note that I don't understand whispers so you might want to write a one or two sentence description of the problem as I head to your seat. I will write corrections on the white board. Bug bounties will be awarded in final exam points.

If you find a minor spelling or grammatical error, please do not disturb everyone else by reporting it. Such errors can be reported to the newsgroup after the exam. If two reports of the same error are posted close enough together that they were plausibly written at the same time, I will give points to both. Points for minor errors will be awarded in homework points.
Good luck!

1. **(13 points):** Powers of Three.
   Describe a TM that accepts the set $\{a^n \mid n \text{ is a power of } 3\}$. You should sketch how the machine works, but you don't need to give a detailed description of its transition function.

2. **(13 points):** Primes.
   Is the set
   $$B = \{M \mid M \text{ accepts iff the length of its input is prime number}\}$$
   recursive, recursively enumerable, neither or both? Give a short justification for your answer.

3. **(13 points):** Leftist Turing Machines.
   Is the set
   $$B = \{M\#x \mid M \text{ moves its head to the left at most 10 times when run with input } x\}$$
   recursive, recursively enumerable, neither or both? Give a short justification for your answer.

4. **(13 points):** Outer Thirds.
   Let $A$ be a language. Define
   $$\mathit{OuterThirds}(A) = \{w \mid \exists x, y, z.(|x| = |y| = |z|) \wedge (w = xz) \wedge xyz \in A\}$$
   Show that there exists a language $A$ such that $A$ is regular and $\mathit{OuterThirds}(A)$ is not. Give a proof with your answer.

5. **(13 points):** Context-Free, or Not?
   Consider the two languages below:
   $$\begin{array}{rcl} B & = & \{w \in \{a,b\}^* \mid \exists i.(w = a^i b^{2i}) \vee (w = a^{2i} b^i)\} \\ \widetilde{B} & = & \sim B \end{array}$$
   (a) **(7 points):** Is $B$ context-free? If it is give a CFG that generates it and a brief explanation of how your CFG works; otherwise, prove that $B$ is not context-free.
   (b) **(6 points):** Is $\widetilde{B}$ context-free? If it is, give a CFG that generates it and a brief explanation of how your CFG works; otherwise, prove that $\widetilde{B}$ is not context-free.

6. **(13 points):** No Exceptions.
   Let x be an array in a Java program. If x has $n$ elements, then the valid values for indices for x are $0 \ldots n-1$. A Java program throws an ArrayIndexOutOfBounds exception if the program attempts to access an array with an index value that is outside of this range.

   For simplicity, we'll assume that the Java programs that we want to analyse have no command line arguments and read all of their input

   Let $A = \{\mathit{src\#input} \mid \mathit{noExcept}(\mathit{src}, \mathit{input})\}$ where $\mathit{noExcept}(\mathit{src}, \mathit{input})$ is true iff the Java program with source-code *src* throws an ArrayIndexOutOfBounds exception when run with input *input*.

   Is the set $A$ recursive, recursively-enumerable, neither, or both? Give a short justification for your answer.

7. **(13 points):** Context-Free Shuffling.
Recall the shuffle operation. If $x, y \in \Sigma^*$, we write $x\|y$ for the set of all strings that can be obtained by shuffling strings $x$ and $y$ together like a deck of cards. Formally,

$$\begin{array}{rcl} \epsilon\|y & = & \{y\} \\ x\|\epsilon & = & \{x\} \\ xa\|yb & = & ((x\|yb)\cdot a) \cup ((xa\|y)\cdot b) \end{array}$$

The shuffle of two languages, $A$ and $B$, denoted $A\|B$, is the set of all strings obtained by shuffling a string from $A$ with a string from $B$:

$$A\|B \quad = \quad \bigcup_{\substack{x \in A \\ y \in B}} x\|y$$

Are the context-free languages closed under the shuffle operation? In other words, if $A$ and $B$ are context-free, is $A\|B$ guaranteed to be context-free? Justify your answer.

8. **(13 points):** Integer Constants in Java.
An integer constant in Java can be an octal constant, a decimal constant, or a hexadecimal constant. An octal constant starts with the character '0' and is followed by zero or more octal digits. A decimal constant starts with the any of the characters, '1', '2', '3', '4', '5', '6', '7', '8', or '9', and is followed by zero or more decimal digits. A hexadecimal constant starts with the string "0x" and is followed by one or more hexadecimal digits. Any of these forms may have the character 'l' (a lower case L) at the end to indicate that it is a long (i.e. 64 bit) constant. An octal digit is any element of the set

$$Oct \quad = \quad \{\text{'0'}, \text{'1'}, \text{'2'}, \text{'3'}, \text{'4'}, \text{'5'}, \text{'6'}, \text{'7'}\}$$

A decimal digit is any element of the set

$$Dec \quad = \quad Oct \cup \{\text{'8'}, \text{'9'}\}$$

A hexadecimal digit is any element of the set

$$Hex \quad = \quad Dec \cup \{\text{'a'}, \text{'b'}, \text{'c'}, \text{'d'}, \text{'e'}, \text{'f'}\}$$

Write a regular expression that matches Java integer constants as defined above (and matches no other strings). You may build up your regular expression from smaller expressions, and you may use $Oct$, $Dec$, and $Hex$ within your expressions.

9. **(13 points):** Parsing Poker.

The POKER programming language has four arithmetic operators: $\clubsuit$, $\diamondsuit$, $\heartsuit$, and $\diamondsuit$ each defined on the non-negative integers as shown below:

$$\begin{array}{rcll} x \clubsuit y & = & x^2 + y^2, & 12 \clubsuit 8 = 208 \\ x \diamondsuit y & = & \mathsf{gcd}(x, y), & 12 \diamondsuit 8 = 4 \\ x \heartsuit y & = & \mathsf{lcm}(x, y), & 12 \heartsuit 8 = 24 \\ x \spadesuit y & = & x^y, & 12 \spadesuit 8 = 429,981,696 \end{array}$$

where gcd and lcm denote the greatest-common-divisor and least-common-multiple respectively. The $\clubsuit$ and $\heartsuit$ operators can also be used as unary operators. The expression $\clubsuit x$ denotes the sum of the positive factors of $x$ (other than $x$ itself) – for example, $\clubsuit 12 = 1 + 2 + 3 + 4 + 6 = 16$, and $\clubsuit 137,438,691,328 = 137,438,691,328$. The expression $x \heartsuit$ denotes the largest prime factor of $x$ – for example, $12 \heartsuit = 3$.

Here's a CFG for POKER:

$$\begin{array}{rcl} expr & \rightarrow & \alpha \mid \clubsuit\, expr \\ \alpha & \rightarrow & \beta \mid \alpha \clubsuit \beta \\ \beta & \rightarrow & \gamma \mid \beta \diamondsuit \gamma \\ \gamma & \rightarrow & \eta \mid \gamma \heartsuit \eta \\ \eta & \rightarrow & \theta \mid \eta \heartsuit \\ \theta & \rightarrow & \mu \mid \theta \spadesuit \mu \\ \mu & \rightarrow & \mathsf{CONST} \mid \mathsf{LPAREN}\ expr\ \mathsf{LPAREN} \end{array}$$

where the terminal symbols are $\clubsuit$, $\diamondsuit$, $\heartsuit$, $\spadesuit$, CONST (an integer constant, written in decimal notation), LPAREN (a left parenthesis), and RPAREN (a right parenthesis).

(a) **(7 points):** Is "$3 \clubsuit 52 \diamondsuit 42 \heartsuit \heartsuit 8$" generated by the CFG for POKER expressions? If so, draw the parse-tree for the expression and determine the value of the expression; otherwise, give a brief explanation (one or two sentences) of why it is not possible to generate this expression.

(b) **(6 points):** Is "$3 \clubsuit \clubsuit 52 \diamondsuit 42 \heartsuit 8$" generated by the CFG for POKER expressions? If so, draw the parse-tree for the expression and determine the value of the expression; otherwise, give a brief explanation (one or two sentences) of why it is not possible to generate this expression.

10. **(13 points):** Functions that Grow Really Fast.
Consider the following functions:

$f$:

$$\begin{array}{rcl} f(n,0) & = & n! \\ f(n,m) & = & f(f(f(f \ldots f(n,m-1)\ldots),m-1),m-1),m-1), \quad n \text{ nested applications of } f(\cdot, m-1). \end{array}$$

If you prefer, here's an equivalent definition of $f$:

```
integer $f$(integer n, integer m) {
  if(m == 0) return(n!);
  else {
    integer q = n;
    for(int i = 0; i < n; i++) q = f(q, m-1);
    return(q);
  }
}
```

For example,

$$\begin{array}{rclcl} f(3,0) & = & 3! & = & 6 \\ f(3,1) & = & f(f(f(3,1),1),1) & = & f(f(3!,1),1) \\ & = & & = & f(f(6,1),1) \\ & = & & = & f(6!,1) \\ & = & & = & f(720,1) \\ & = & & = & 720! \\ & = & & \approx & 2.60121894 * 10^{1746} \end{array}$$

To calculate $f(3,2)$, we start with 720! – let $X = 720!$. We then calculate $X!!!\ldots!$, with 720! (roughly $2.6 * 10^{1746}$) factorial operations total. Let that number be $Y$. We now calculate $Y!!!\ldots!$ with $Y$ factorial operations. If you think this number is big, note that it's only $f(3,2)$. The number for $f(3,3)$ is way, way bigger. In fact, if we tried to write it as $10^{10^{10^{\cdot^{\cdot}}}}$, we would need a more exponents in the stack than the number of atoms in the universe.

$g$: Let $g(n) = f(n,n)$. For example,

$$\begin{array}{rcl} g(0) & = & 1 \\ g(1) & = & 1 \\ g(2) & = & 2 \\ g(3) & = & f(3,3), \quad \text{(really, \textbf{REALLY} big, see comments above)} \end{array}$$

$h$: Let

$$HP_n = \{M\#x \in HP \mid |M\#x| \leq n\}$$

where $M\#x \in HP$ iff Turing machine $M$ halts when run on input $x$. Define

$$\begin{array}{rcll} h(n) & = & \max_{M\#x \in HP_n} nsteps(M,x), & \text{if } HP_n \neq \varnothing \\ & = & -1, & \text{otherwise} \end{array}$$

where $nsteps(M,x)$ is the number of steps the $M$ takes before halting when run with input $x$.

We say that function $F_1$ *dominates* function $F_2$ iff there exists some integer $m$ such that for all $n \geq m$, $F_1(n) > F_2(n)$.

(a) **(7 points):** Does $g$ dominate $h$?

(b) **(6 points):** Does $h$ dominate $g$?

Give a short justification for each answer.