CMSC330 Fall 2016 Midterm #2
2:00pm/3:30pm

Gradescope ID: | | | | | | |

(Gradescope ID is the First letter of your last name and last 5 digits of your UID)

(If you write your name on the test, or your gradescope ID is not correct, your test will NOT be graded)

Discussion Time:	**10am**	**11am**	**12pm**	**1pm**	**2pm**	**3pm**
TA Name (Circle):	**Alex**	**Austin**	**Ayman**	**Brian**	**Damien**	**Daniel K.**
	Daniel P.	**Greg**	**Tammy**	**Tim**	**Vitung**	**Will K.**

Instructions

1. Do not start this test until you are told to do so!
2. You have 75 minutes to take this midterm.
3. This exam has a total of 100 points, so allocate 45 seconds for each point.
4. This is a closed book exam. No notes or other aids are allowed.
5. Answer short answer questions concisely in one or two sentences.
6. For partial credit, show all of your work and clearly indicate your answers.
7. Write neatly. Credit cannot be given for illegible answers.

	Problem	Score
1	Finite Automata	/20
2	Context Free Grammars	/16
3	Parsing	/8
4	OCaml	/20
5	Programming Language Concepts	/12
6	Operational Semantics	/10
7	Lambda Calculus	/14
	Total	/100

GRADESCOPE ID:

1. Finite Automata (20 pts)

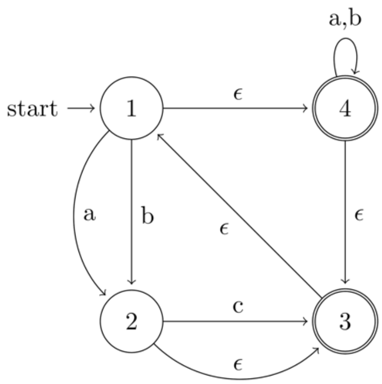


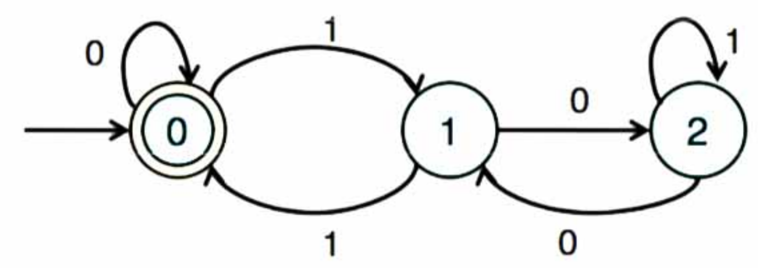
Figure 1: NFA

A. (5 pts) Convert the NFA in Figure 1 to a DFA.

B. (5 pts) Write a regular expression that accepts the same language as the NFA shown in Figure 1.

C. (5 pts) Construct a DFA or NFA that accepts the set of strings over {a, b} that contain exactly two b's.

D. (5 pts) Reduce the following DFA to Regular Expression. (Hint: delete states 2, 1, and 0 in that order.)



2. Context Free Grammars (16 pts)

A. (3 pts) The following context-free grammar generates all strings of the form $a^x b^y$ for some x and y:

```
S -> aT
T -> aTbb|ε
```

Define an equation relating x and y.

B. (6 pts) Give a context-free grammar for binary number expressions involving &, +, and ~ (and, or, not)

1) Order of precedence (highest to lowest): ~, &, +
2) Associativity for & and + is left, ~ is unary
3) Parenthesis should be supported
4) Binary numbers can start with 0's (1 and 01 are valid)

C. (4 pts) Let G be the context-free grammar

$S \rightarrow \mathrm{a}S \mid S\mathrm{b} \mid SS \mid \mathrm{ab}$

a) Give a regular expression for the language of G.

b) Prove that G is ambiguous. Give an unambiguous grammar that generates the same language as G.

D. (3 pts) Give a context-free grammar G that generates the language $L = L_1 \cup L_2$, where
$L_1 = \{a^n b^n c^m : n, m > 0\}$
$L_2 = \{a^n b^m c^m : n, m > 0\}$

3. Parsing (8 pts)

A. (3 pts) Consider the following context-free grammar:

S → (A) | a
A → S B
B → ; S B | ε

Compute the first sets of each non-terminal

FIRST(S) = { }
FIRST(A) = { }
FIRST(B) = { }

B. (5 pts) Consider the following CFG.

S → E; S | E
E → T O E | T
T → n
O → + | -

Draw a parse tree for the string "n+n;n-n"

4. OCaml (20 pts)

```
let rec map f l = match l with
     [] -> []
     | h::t -> let r = f h in r :: map f t
;;
let rec fold f a l = match l with
    [] -> a
   |h::t -> fold f (f a h) t
;;
helper methods are allowed.
```

A. (8 points) Write a function **combine** that, given a list of potentially overlapping intervals (pairs) sorted by the first element of each pair, return a list with the overlapping pairs from the original list having been combined.

```
combine [(1,3);(3,5);(7,9)] = [(1,5);(7,9)]
combine [(1,3);(4,8);(5,9)] = [(1,3);(4,9)]
```

let rec combine lst =

B. (8 points) Write a function **rotate** that, given a list and a positive integer k, rotate the list k elements to the left. **You are alloawed to use "@" to merge lists.**

rotate 2 [1;2;3;4;5] = [3;4;5;1;2]
rotate 0 [1;2;3;4;5] = [1;2;3;4;5]
rotate 8 [1;2;3;4;5] = [3;4;5;1;2]

```
let rec rotate lst =
```

C. (4 points) Using **fold** and/or **map**, write a function to compute the sum of squares each item in a list of **floating point** values.

square_sum [1.5;4.0;2.0] = 22.25
square_sum [1.0;2.0;3.0] = 14.00

```
let rec square_sum lst =
```

5. Programming Language Concepts (12 pts)

A. (4 pts) The following code calculates 1+2+3+4…n. It is not tail recursive. Rewrite the function sum, so that it is tail recursive. (You are allowed to add nested helper functions)

```
    let rec sum n = if n=1 then 1 else n + sum(n-1)
let rec sum n =
```

B. (2 pts) Circle **all** of the statements that apply to the following OCaml pseudocode:

```
    let x = E1 in let x = E2 in E3
```

a) In the scope of E1, x is bound
b) In the scope of E2, x is bound
c) The pseudocode contains an instance of shadowing
d) The code is invalid, as there are multiple declarations of x

C. (1 pts) Which of the following statements is **NOT** true about a language with first-class functions?

a) Functions can be passed in as arguments to other functions
b) Functions can be returned as the result of calling other functions
c) The language does not include imperative features
d) The language treats functions on the same level as other data types

D. (1 pts) The fixed-point combinator is used to achieve _____________________.

E. (1 pts) _________________ type checking occurs during a program's compilation.

F. (1 pts) True / False   Statically typed languages are type safe.

G. (2 pts) True / False   I voted.

6. Operational Semantics (10 pts)

Use the language defined by the context-free grammar

$e \to n \mid x \mid e \texttt{ + } n \mid \texttt{let } x \texttt{ = } e \texttt{ in } e$

$n \to$ decimal integers

$x \to$ alphabetic variable names

and the operational semantics

$$\text{Number: } \frac{}{A; n \to n} \qquad \text{Sum: } \frac{A; e_1 \to n_1 \qquad A; e_2 \to n_2}{A; e_1 \texttt{ + } e_2 \to n_1 + n_2}$$

$$\text{Var: } \frac{A(x) = v}{A; x \to v} \qquad \text{Let: } \frac{A; e_1 \to v_1 \qquad A, x : v_1; e_2 \to v_2}{A; \texttt{let } x \texttt{ = } e_1 \texttt{ in } e_2 \to v_2}$$

to fill in the blank boxes in the following two derivations:

$$\dfrac{\dfrac{\dfrac{}{\bullet; \texttt{1} \to \Box} \qquad \dfrac{}{\bullet; \texttt{2} \to \Box}}{\bullet; \texttt{1 + 2} \to \Box} \qquad \dfrac{}{\bullet; \texttt{3} \to \Box}}{\bullet; \texttt{1 + 2 + 3} \to \Box}$$

$$\dfrac{\dfrac{}{\bullet; \texttt{10} \to \mathit{10}} \qquad \dfrac{\dfrac{}{\Box; \texttt{20} \to \mathit{20}} \qquad \dfrac{\dfrac{}{\Box; \texttt{x} \to \mathit{20}} \qquad \dfrac{}{\Box; \texttt{y} \to \mathit{10}}}{\Box; \texttt{x + y} \to \mathit{30}}}{\Box; \texttt{let x = 20 in x + y} \to \mathit{30}}}{\bullet; \texttt{let y = 10 in let x = 20 in x + y} \to \mathit{30}}$$

7. Lambda Calculus (14 pts)

A. Alpha Equivalence (2 point each): For each pair of expressions, determine if they are alpha equivalent. Circle "Equivalent" or "Not Equivalent".

λf. λx. f (f x y) y
λf. λx. f (f x z) z

Equivalent　　Not Equivalent

λa. λb. (λa. a b) (λb. b b) a
λa. λx. (λa. a x) (λb. x x) a

Equivalent　　Not Equivalent

B. Beta Reduction: Reduce each expression to normal form. If it reduces infinitely, state that it does not reduce.

(2 points) λx. (λz. z z x) a b

(3 points) (λx. λy. y x y) (a b) (λm. a m)

C. Operations on Church Numerals (5 points): Given the following definitions, prove that mult 2 * 0 = 0. Show all of your beta reductions and alpha conversions for full points.

0 = λf.λy. y　　　　2 = λf.λy. f (f y)
mult =λM. λN. λ f. M (N f)

Prove mult 2 * 0 = 0