CMSC330 - Organization of Programming Languages
Summer 2023 - Exam 1

CMSC330 Course Staff
University of Maryland
Department of Computer Science

Name: ______________________________

UID: ______________________________

I pledge on my honor that I have not given or received any unauthorized assistance on this assignment/examination

Signature: ______________________________

Ground Rules

- You may use anything on the accompanying reference sheet anywhere on this exam
- Please write legibly. **If we cannot read your answer you will not receive credit**
- You may not leave the room or hand in your exam within the last 10 minutes of the exam
- If anything is unclear, ask a proctor. If you are still confused, write down your assumptions in the margin

Question	Points
Q1	10
Q2	15
Q3	15
Q4	15
Q5	20
Q6	15
Q7	10
Total	100

Problem 1: Language Concepts

[Total 10 pts]

	True	False
Any regular expression can be expressed as a Context Free Grammar any set of strings a RE can construct, a CFG can too	● T	○ F
`let f x = x 4` is an example of a higher order function f is a function that takes in another function	● T	○ F
One could theoretically code project 1 in lambda calculus it is a turing complete language, and project 1 is solveable	● T	○ F
All statically typed languages use explicit (manifest) typing Ocaml uses implicit typing but is also statically typed	○ T	● F
FSMs are a subset of Turing Machines in terms of computational power FSM can solve certain types of problems. TM can solve any solveable problem	● T	○ F

Problem 2: Typing

[Total 15 pts]

Write an expression of the following types in OCaml. You cannot use type annotations, and all pattern matching must be exhaustive.

(a) `string -> ’a -> string` [2 pts]

fun x y -> x ˆ "hello" (If you do not use the second parameter, it becomes generic)

(b) `’a -> ’a -> bool -> ’a` [3 pts]

fun x y z -> if z then x else y (z must be a bool and x and y must be the same type)

Given the following OCaml expressions, write down its type.

(c) `fun a b -> let c = a = b in if c then 2 else 3` [2 pts]

’a -> ’a -> int (a and b are being compared and an int is being returned)

(d) `fun a b c d -> if a && let x = b > c in x then d + 1 else b` [3 pts]

bool -> int -> int -> int -> int (b and d must be ints, and b is being compared to c)

(e) Which of the following choices could be the type of the python lambda below? Select all that apply. [2 pts]

`lambda x,y: x + y`

● A int -> int -> int ○ B string -> int -> string ● C list -> list -> list ● D float -> int -> float

○ E None of the above you can use the + operator on lists, floats and ints

(f) Which of the following python lambdas could have the type of string list -> int list? Select all the apply. [3 pts]

● A `lambda x: [1,2] if x == ["hello"] else [0]`
○ B `lambda x: [len(x[0])]`
○ C `lambda x: map(lambda y:  len(y),x)`
○ D `lambda x: len(x)`
○ E None of the above

C returns map object, D does not return a list

Problem 3: Regular Expressions [Total 15 pts]

(a) Which of the following strings are an exact match of the following Regular Expression? Mark all that apply. [5 pts]

```
^[A-Z][a-z0-9]+: ([0-9]{3}|[CS330]+)$
```

● A Major: CS   ○ B Age: 25   ● C Class: CS330   ● D Finitial: C   ○ E None

(b) Write a regular expression that accepts phone numbers of all the following formats and rejects everything else. You may assume that any X can be any digit. [5 pts]

XXX-XXX-XXXX   XXX-XXXXXXX   XXXXXXXXXX   (XXX)-XXX-XXXX   (XXX)-XXXXXXX   (XXX)XXXXXXX

```
((\d{3})|\d{3})((-\d{3}-?\d{4})|\d{7})
```

(c) Write a regular expression that would accept all strings of odd length and have at least 1 lowercase vowel (a,e,i,o,u) and reject anything else [5 pts]

```
(..)*[aeiou](..)*|(..)*.[aeiou].(..)*
```

Problem 4: Context Free Grammars [Total 15 pts]

Consider the following Grammars:

Grammar 1	Grammar 2	Grammar 3	Grammar 4
S -> AB	S -> ASB|a	S -> Sc|AB	S -> ASB|cScc|c
A -> aAa|a	A -> aA|a	A -> aA|a	A -> aaA|a
B -> bBbb|ϵ	B -> bbB|c	B -> bbB|b	B -> bbB|b

(a) Which grammars (of 1, 2, and 3) accept both "aabbbbc" and "aaabbcc"? Select all that apply. [4 pts]

○ 1 Grammar 1   ○ 2 Grammar 2   ○ 3 Grammar 3   ● N None

(b) Ambiguity [6 pts]

	Yes	No
"aaabbb" is an ambiguous string in Grammar 1	○ Y	● N
"aaabbc" is an ambiguous string in Grammar 2	○ Y	● N
"aaabcc" is an ambiguous string in Grammar 3	○ Y	● N

(c) Which strings are accepted by Grammar 4? Select all that apply. [5 pts]

● A aaacbbb   ○ B aaacbbbb   ○ C ccaaabbbbcc   ○ D cacacbbbb   ○ E None

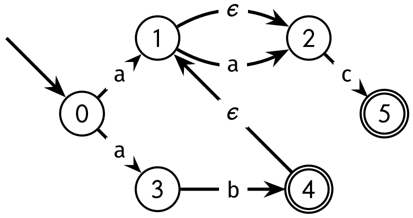
Problem 5: Finite State Machines

[Total 20 pts]

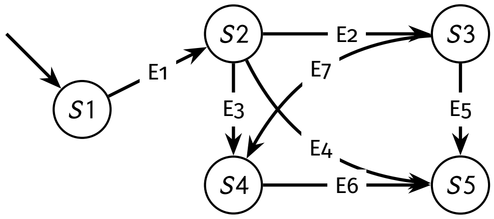
(a) Using the subset algorithm, convert the following NFA to a DFA, and fill in the blanks appropriately matching the DFA provided with the right nodes and transitions. Only the blanks will be graded. [12 pts]

NFA:

Scratch Space (if needed)



DFA:



S1:	0	S2:	[1,2,3]	S3:	[1,2,4]	S4:	2
S5:	5	E1:	a	E2:	b	E3:	a
E4:	c	E5:	c	E6:	c	E7:	a

(b) Which of the following are the final states? Select all that apply [3 pts]

① S1 ② S2 ❸ S3 ④ S4 ❺ S5 Ⓝ None

(c) Write a regex to describe the language of the above NFA [5 pts]

$ab|ab?a?c$

Problem 6: Lambda Calculus

[Total 15 pts]

For the following questions perform a single β-reduction using eager (call by value) evaluation on the outermost expression. If you cannot reduce it, write **Beta Normal Form**. You may **not** α-convert your final answer.

(a) $(\lambda y.yy)((\lambda x.y)(\lambda y.xy))$ [2 pts]

$(\lambda y.\ y\ y)\ y$

(b) $(\lambda x.\lambda x.xx)(z\ (\lambda a.a))$ [3 pts]

$\lambda x.\ x\ x$

For the following questions perform a single β-reduction using lazy (call by name) evaluation on the outermost expression. If you cannot reduce it, write **Beta Normal Form**. You may **not** α-convert your final answer.

(c) $(\lambda y.yy)((\lambda x.y)(\lambda y.xy))$ [2 pts]

$((\lambda x.y)(\lambda y.xy))((\lambda x.y)(\lambda y.xy))$

(d) $(\lambda x.\lambda x.xx)(z\ (\lambda a.a))$ [3 pts]

$\lambda x.\ x\ x$

(e) Which of the following is alpha equivalent to $(\lambda x.x\lambda x.xy)$? Select all that apply. [2 pts]

○ A $(\lambda z.z\lambda x.zy)$ ○ B $(\lambda y.y\lambda x.xy)$ ● C $(\lambda z.z\lambda x.xy)$ ○ D $(\lambda x.x\lambda y.yz)$ ○ G None

(f) Convert the following to Beta Normal Form: $(\lambda z.\lambda x.xz)(\lambda y.yy)c$ [3 pts]

○ A c ○ B $(\lambda x.x\ x)c$ ● C $c\ (\lambda y.y\ y)$ ○ D $\lambda x.x\ (c\ c)$ ○ E $c\ c$ ○ F Infinite Recursion ○ G None

Problem 7: Python Programming

[Total 10 pts]

(a) Write a function mur that has the same functionality of map, but uses reduce. [4 pts]

```
def mur(f,lst):
    return reduce(___BLANK____)

#mur(lambda x: x + 1,[1,2,3]) => [2,3,4]
#mur(lambda x: len(x),[[1,2,3],[4,5],[6]]) => [3,2,1]
#mur(lambda x: x,[1,2,3]) => [1,2,3]
```

Blank:

lambda a h: a + [f(h)], lst, []

(b) Write a function sumnum that takes in a formatted string and returns the sum of all the numbers found in that string. [6 pts]

```
#sumnum("I have 2 apples and 30 oranges") => 32
#sumnum("There are no numbers here") => 0
#sumnum("I can have negatives like -2 and -4") => -6

def sumnum(s):
   return sum(map(lambda x:  float(x), re.findall(r"-?[0-9]+"))
```