

NAME (PRINT): \_\_\_\_\_

STUDENT NUMBER (PRINT): \_\_\_\_\_

University of Toronto Mississauga  
FALL 2024 MOCK FINAL EXAMINATION  
Introduction to Theory Computation

**Macho Man ( $m^2$ )**

**Duration - 3 hours 10 minutes**

**Aids : お前のお母さん:**

[https://youtu.be/3UHR\\_XWrGhc](https://youtu.be/3UHR_XWrGhc)

*The University of Toronto Mississauga and you, as a student, share a commitment to academic integrity. You are reminded that you may be charged with an academic offence for possessing any unauthorized aids during the writing of an exam. Clear, sealable, plastic bags have been provided for all electronic devices with storage, including but not limited to: cell phones, smart watches, SMART devices, tablets, laptops, and calculators. Please turn off all devices, seal them in the bag provided, and place the bag under your desk for the duration of the examination. You will not be able to touch the bag or its contents until the exam is over.*

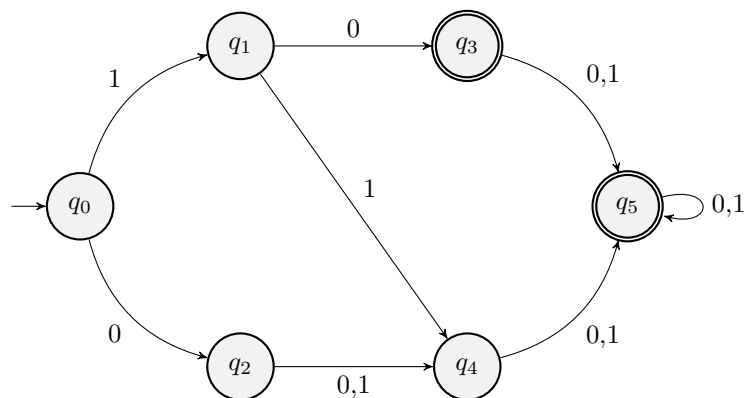
*If, during an exam, any of these items are found on your person or in the area of your desk other than in the clear, sealable, plastic bag, you may be charged with an academic offence. A typical penalty for an academic offence may cause you to fail the course.*

*Please note, once this exam has begun, you **CANNOT** re-write it.*

This is a mock exam designed for studying CSC236. Any and all similarities with the Fall 2024 CSC236 final examination are purely coincidence.

This examination is meant to be done in 3 hours, but it is a little bit long, so do not worry about finishing. There are a total of 9 questions worth a total of 78 points. Answer the questions in the space provided.

**Q1. (9 points)** Consider the DFA below:



- a) (1 point) Describe the language accepted by this DFA.
- b) (3 points) Convert this DFA into a minimal NFA (i.e., there is no smaller NFA that accepts this language). Give a brief justification for why it works.

c) (2 points) Provide a DFA that accepts the language matched by  $(a+ab)^*$ , where  $\Sigma = \{a, b\}$ .

d) (3 points) Give a state invariant for each state in your DFA. You do not need to prove them.

**Q2. (10 points)** For each of the statements below, decide whether it is **true or false** and provide a proof justifying your answer.

a) (2 points) Let  $\Sigma = \{0, 1\}$ . The language  $L = \{w \in \Sigma^* : |w| = 2\}$  is regular.

b) (2 points) Let  $\Sigma = \{0, 1\}$ .  $L = \{1^i 0^j : i, j \in \mathbb{N}, i \geq j\}$  is a regular language.

c) (2 points) Let  $L, M$  be regular languages. The language  $L \cap M$  is regular.

d) (2 points) Let  $\Sigma = \{a, b, c\}$ . The language of all strings whose characters are alphabetically ordered is regular. (Note: The empty string is considered alphabetically ordered)

e) (2 points) Let  $L, M$  be languages. If  $L$  is not regular, then  $L \cup M$  is not regular.

**Q3. (8 points)**

- a) (4 points) State the CLRS version of master theorem. Define all variables and state their restrictions.

- b) (4 points) Let  $f : \mathbb{N} \rightarrow \mathbb{R}$  be a nonnegative function. Prove that if  $f \in \Theta(n^k)$  for some  $k > 0$ , then the regularity condition holds true.

**Q4. (7 points)** Consider the program below:

```
1      def binary_search(x: int, lst: list[int]):
2          l ← 0, r ← len(lst) - 1
3          while(r - l > 0):
4              mid ← (l+r) // 2
5              if(lst[mid] == x):
6                  return mid
7              elif(lst[mid] < x):
8                  r ← mid
9              else:
10                 l ← mid + 1
11         if(lst[l] == x):
12             return l
13         return -1
```

- a) (2 points) State the preconditions and postconditions of this program.
- b) (5 points) Prove that this program is correct.

**Q5. (6 points)** Let  $A$  be a set of functions defined recursively as follows:

- $\sqrt{x} \in A$
- If  $f \in A$ , then  $\frac{1}{f} - f \in A$

a) (4 points) Let  $P(f)$  be a predicate on  $A$  and suppose you have managed to prove that

- $P(\sqrt{x})$  is true,
- $P(f) \implies P(\frac{1}{f} - f)$  for all  $f \in A$ .

Prove that  $\forall f \in A, P(f)$ . You may only assume that the principle of induction holds for the natural numbers, NOT the set  $A$ .

b) (2 points) Prove that  $\forall f \in A, f\left(\frac{1}{2}\right) = \frac{1}{\sqrt{2}}$ .



**Q6. (7 points)** Let  $\Sigma = \{0, 1\}$ . For any language  $L \in \Sigma^*$ , define

$$S = \{w \in \Sigma^* : 0w1 \in L\}$$

Given that  $L$  is a regular language, prove that  $S$  is a regular language.

**Q7. (8 points)** Recall that a segment tree is a data structure effective for querying and updating information about a range of values in a list, such as the minimum element in a provided range. In fact, we will examine an implementation of a segment tree that does this.

Suppose that we have a list of numbers `lst`. In this implementation, the segment tree can be thought of as a binary tree, with each node storing a range of indices  $[l, r]$ , a value which represents the minimum number in `lst[l:r]` (inclusive of both sides), and pointers to a left and right child. **Note that a node does not have a left and right child if and only if it is true that  $l = r$ .** Otherwise, the left child will keep track of the range  $[l, \lfloor \frac{l+r}{2} \rfloor]$  and the right child will track  $[\lfloor \frac{l+r}{2} \rfloor + 1, r]$

As an example, consider the following pseudocode for `update`, which updates the segment tree to correctly return queries about `lst` after setting `lst[i] = x`:

```

1      def update(root: Node, x: int, i: int):
2          if i not in [root.l, root.r] then return
3          else if root.l = i = root.r then
4              root.val ← x
5              return
6          update(root.left_child, x, i)
7          update(root.right_child, x, i)
8          root.val ← min(root.left_child.val, root.right_child.val)

```

a) (3 points) Prove that this method is correct.

b) (3 points) Find the recursive worst-case runtime of `update`.

c) (2 points) Find the tight asymptotic bound for the runtime of `update`.

**Q8. (15 points)** Recall the statement of the *pumping lemma*:

Let  $L$  be a regular language associated with some alphabet  $\Sigma$ . Then  $\exists p \in \mathbb{N}^+, \forall w \in L$ , where  $|w| \geq p$ ,  $\exists x, y, z \in \Sigma^*$  so that

- $|y| \geq 1$
- $|xy| \leq p$
- $\forall n \in \mathbb{N}, xy^n z \in L$

If a language satisfies the statement in the pumping lemma, we say that it can be **pumped**.

- a) (3 points) Show why the statement must have the condition  $|xy| \leq p$ . (Hint: Consider the language  $\{w \in \{a, b\}^* : w = a^i b^j, i \leq j\}$ )

- b) (2 points) Find with proof the value of  $p$  for the language  $\mathcal{L}(((ab)^*a)^*)$

- c) (1 point) Explain why every finite language can be pumped.

Now, assume that  $L, M$  are regular languages that can be pumped.

d) (2 points) Prove that  $L \cup M$  can also be pumped.

e) (2 points) Prove that  $M^*$  can be pumped (You would want to consider when  $M$  is finite or when  $M$  is infinite).

f) (3 points) Prove that  $LM$  can be pumped.

g) (2 points) Prove the pumping lemma.

**Q9. (10 points)** Given two bits of either 0 or 1, the XOR operator returns 1 if and only if exactly one of the bits is 1. This operator can be generalised to be used with binary strings by comparing each individual bit. It is denoted by the symbol  $\wedge$ . When given integer inputs, the operator will automatically convert them to binary, perform the operation, and output the result as a decimal number. This operator can be quite useful, as it will be illustrated in this question.

Consider a list with with the property that there is only one number whose number of occurrences in the list is odd. You can also think of it as every number having an even number of occurrences except for one number. Take a look at the method below:

```

1      def find_odd_number(A):
2          '''
3          -----Pre:-A is a list with x being the only number having an
4          -----odd number of occurrences
5          -----Post:-Returns the odd element x
6          -----'''
7          m ← 0
8          i ← 0
9          while i < len(lst):
10             m ← m ^ lst[i]
11             i ← i + 1
12          return m

```

Prove the correctness of this program. You may use the fact that the XOR operator is commutative.