- This assignment is **due in Week 10 on Sunday 15 October, 11:59pm**.
- Submission is on GradeScope.
  - For problems 1 and 2, you will submit one PDF with your solutions.
  - For problem 3, you will submit your TMs in files named `problem_3_1.txt`, `problem_3_2.txt`, `problem_3_3.txt`.
  - For problem 4, you will submit files named `run.sh`, `build.sh` and other files you require. The first argument of `run.sh`, either `tm_to_ptm` or `ptm_to_tm`, should choose the solution that is run. A skeleton will be provided on Ed.
  - For problem 5, you will submit your 2-tape TM in one file named `problem_5.txt`.
- All work must be **done individually** without consulting anyone else's solutions in accordance with the University's "Academic Dishonesty and Plagiarism" policies.
- For clarifications and more details on all aspects of this assignment (e.g., level of justification expected, late penalties, repeated submissions, what to do if you are stuck, etc.) you are expected to regularly monitor the Ed Forum post "Assignment FAQ".

DTM stands for "Deterministic Turing Machine", and NTM stands for "Nondeterministic Turing Machine".

All tapes in this assignment are doubly-infinite. When asked to give a **low-level description** use Morphett's format, and its extension to 2 tapes as follows: the instruction

`q a b c d L R s`

means that if the TM is in state q and reads a on the first tape and b on the second tape then it writes c on the first tape and d on the second, and moves the first tape left and the second tape right, and changes to state s.

For instance, suppose the question asked you to provide a 2-tape DTM that checks if a string has the same number of 0s as 1s. Here is an answer:

```
; copy0: copies 0's from tape#1 to tape#2
copy0 0 _ 0 0 R R copy0
copy0 1 _ * * R * copy0
copy0 _ _ * * L L compare

; compare: matches 1s on tape#1 with 0s on tape#2
compare 1 0 * * L L compare
compare 0 * * * L * compare

; same number
compare _ _ * * * * halt-accept

; more 1s than 0s
compare * _ * * * * halt-reject

; more 0s than 1s
compare _ * * * * * halt-reject
```

NB. The Morphett's simulator only correctly simulates deterministic TMs. For nondeterministic TMs it resolves nondeterminism randomly, which is not what TMs do! Thus, you cannot rely on the simulator showing you all possible runs on a given input.

Problem 1. (20 marks)

1. (5 marks) Is the complement of every context-free language Turing-decidable? Give a brief justification of your answer.

2. (5 marks) Suppose that L is the intersection of a context-free language L_1 and a regular language L_2. Argue that L is in **P**. You may use any facts proven in the lecture slides.

3. (5 marks) Fix $\Sigma = \{0, 1\}$. Consider the operation dub that maps a string u to the string in which every 0 is replaced by 00. For a language L let $dub(L) = \{dub(u) : u \in L\}$.

   For instance, if $L = \{10, 1001, 11, \epsilon\}$ then $dub(L) = \{100, 100001, 11, \epsilon\}$, and if $L = \{0^n 1^n : n \geq 0\}$ then $dub(L) = \{0^{2n} 1^n : n \geq 0\}$.

   Suppose that L is decidable. Give a high-level description of a decider for $dub(L)$.

4. (5 marks) Consider the language L consisting of the set of strings $(\text{Source}_{M_1}, \text{Source}_{M_2}, \text{Source}_{M_3})$ such that $L(M_1)$ is not regular, $L(M_2)$ is not regular, and $L(M_1) \cup L(M_2) = L(M_3)$.

   Show that L is undecidable. You may only use facts from the lecture slides to do so, e.g., that the acceptance problem TMs is undecidable.

Problem 2. (10 marks) Below is a NTM over input alphabet $\{a, b\}$.

1. (5 marks) Describe in one sentence the language that it accepts. Briefly justify your answer. No marks will be awarded for describing how the TM operates.

2. (5 marks) What is the asymptotic time complexity (aka running time) of the NTM? Give your answer in big-Theta notation, and briefly explain your answer.

```
; initial state is 0
; input alphabet is {a,b}

0 a a R 0
0 b b R 0
0 a A L 1
```

```
1 * * L 1
1 _ _ R 2

2 A A R 4

2 a _ R 2a
2a * * R 2a
2a A A R 2A
2A A A R 2A
2A a A L 1

2 b _ R 2b
2b * * R 2b
2b A A R 2B
2B A A R 2B
2B b A L 1

4 A A R 4
4 _ * * halt-accept
```


Problem 3. (20 marks) Fix $\Sigma = \{0, 1\}$.

1. (5 marks) Give a low-level implementation (in Morphett's format) of a 1-tape DTM that decides the language consisting of strings of the form $x1y$ where $|x| = |y|$. For instance, $1, 011, 10100$ should be accepted, while $\epsilon, 0, 00, 000, 11, 11000$ should be rejected.
2. (5 marks) Give a low-level implementation (in Morphett's format) of a 1-tape DTM that decides the language consisting of strings of the form $x1x$. For instance, $1, 010, 10110$ should be accepted, while $\epsilon, 0, 00, 000, 11, 10101$ should be rejected.
3. (10 marks) Give a low-level implementation (in Morphett's format) of a 1-tape DTM that decides the language consisting of strings of the form $xyxxz$ where x is non-empty. For instance, $1011, 000, 01110110110, 1001000010101$ should be accepted, while $\epsilon, 0, 1101, 01101101, 0011101$ should be rejected.

Problem 4. (20 marks) Following recent advances in interdimensional technology, a new type of Turing Machine has been developed, the *portal Turing Machine* (PTM). These machines have the remarkable power to use 0 symbols on the tape as portals, allowing them to traverse unbounded distances in a single step. (0 looks like a portal, you see.)

The semantics of a PTM are that when a machine is in a cell with a 0 and moves left or right, instead of moving one cell, it moves to the next 0 in that direction.

If there is no 0 in that direction, it "wraps around" the tape and moves to the 0 furthest in the other direction (eg. from the leftmost 0, a "move left" instruction moves to the rightmost 0). If there is only one 0 on the entire tape, a move left or move right instruction causes it to stay put. Note that in each step, the write instruction is executed before the movement instruction (since this may affect whether a 0 is present when moving).

As a high ranking member of the Church of Turing, you want to uphold your Thesis by showing that these new devices are equivalent in computational power to traditional TMs.

1. (10 marks) Write a program to convert a basic TM into an equivalent PTM.
2. (10 marks) Write a program to convert a PTM into an equivalent basic TM.

Input will be a machine with input alphabet $\{0,1\}$, tape alphabet $\{0,1,\_,2,3\}$, all states other than the two halting-states ("halt-reject" and "halt-accept") are of the form $0,1,2,\cdots,9$ (i.e. states are single digit numbers and there are at most 10 non-halting states). Your output should be a machine with the same input alphabet. The tape alphabet may be larger. Input and output machines are both expected in Mogphet's format. Particularly, your output should start with the state named 0 (zero).

For full marks, your simulating machines should have at most 100 states and be reasonably efficient. Public tests and guidelines will be provided to judge this efficiency.

Problem 5. (20 marks)

Build a 2-tape DTM S that takes as input an automaton A over input alphabet $\{0,1,\#\}$ on tape 1, and a string $u \in \{0,1\}^+$ on tape 2, and accepts if $u \in L(A)$, and rejects otherwise. The TM S should run in polynomial time.

The input string u is guaranteed to be non-empty, but is otherwise an arbitrary string the alphabet $\{0,1\}$.

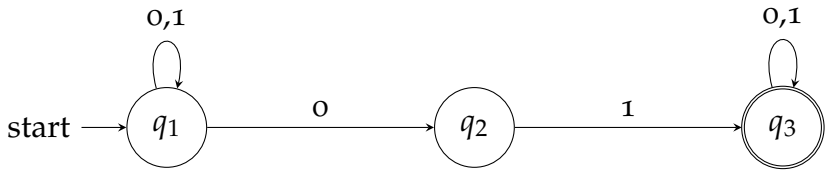
The states of our automata are denoted $q_1, q_2, \cdots, q_n$ for some n. The initial state is q_1, and q_n is the only final state (for simplicity). The automaton has no epsilon transitions. Such an automaton will be represented as a string over input alphabet $\{0,1,\#\}$ as follows. The string encoding the automaton is of the form $u_1u_2\ldots u_n$ where each u_i is of the form $\#v_{i,1}\#v_{i,2}\#\ldots\#v_{i,n}\#$ where each $v_{i,j} \in \{00,01,10,11\}$ is defined as follows: the first bit of $v_{i,j}$ is 1 iff the automaton has a transition from q_i to q_j on symbol 0, and the second bit of $v_{i,j}$ is 1 iff the automaton has a transition from q_i to q_j on symbol 1.

You can assume that the input string to your TM is an encoding of an automaton, i.e., the string matches the regular expression $(\#((0|1)^2\#)^n)^n$ for some n. Subject to this syntactic requirement, the automaton is otherwise arbitrary.

For example, the string

#11#10#00##00#00#01##00#00#11#

encodes the following NFA:



For full marks, your TM should have at most 100 states and be reasonably efficient. Public tests and guidelines will be provided to judge this efficiency.

Your TM should be written in the extension of Morphett's format to 2-tape machines (described above).

10 marks are when A is a DFA, and 10 marks are when A is an NFA.