**CS421 - Yoshii - EX C (based on Weeks 14 -15) (Use black for answers)**

**Turing Machines and Summary**

**=================================================================**

**DUE: Week 15 A in class 🡺 CC on Week15 B No late work is accepted! One per group!!!**

**TOTAL: 22 pts**

**\*\* Group #: 12**

**List names only if present and worked on it.**

**\*\* Name1: Ryan Santos**

**\*\* Name2: Hugh O’Neill**

**\*\* Name3: Qian Zhu**

**===================================================================**

**Be sure to read the context of each question first and give your answer**

**in the format requested.**

**===================================================================**

**===============================================**

**Week14 Inter Question on TM [2pts per Trs = 4 pts]**

**===============================================**

**week14a:\*Inter\* The following Trs' are for Example TM 3 (Subtraction):**

**In q0 - replace 0 in M with blank and enter q1, move R**

**1 Trs Trs(q0, 0) = (q1,blank,R)**

**In q1 - looking for the separation marker (1) and enter q2, move R**

**2 Trs Hint: what do you need to skip over?**

**Trs(q1, 0) = (q1,0,R) // skip over 0’s in M**

**Trs(q1, 1) = (q2,1,R) // into the N section**

**In q2 - look for 0 in N and replace with 1, enter q3, move L**

**2 Trs Hint: what do you need for skip over?**

**Trs(q2,1) = (q2,1,R) // skip over 1’s in N**

**Trs(q2,0) = (q3,1,L) // change 0 to 1**

**Do the rest in the same format, giving the Trs and completing the comments.**

**In q3 - looking for the blank in M, enter q0, move R (U-turn)**

**3 Trs Hint: what do you need to skip over?**

**Trs(q3, 0) = (q3, 0, L) // skip over 0 as moving left**

**Trs(q3, 1) = (q3, 1, L) // skip over 1 as moving left**

**Trs(q3, blank) = (q0, blank, R) // found a blank so U-turn**

**=============================================**

**Week14b - Week 15 Questions on Computability [4 pts]**

**=============================================**

**Q1: If the language you have is recursive**

**then your TM acceptor is an algorithm but**

**If the language you have is recursively enumerable**

**then your TM acceptor is a procedure**

**Q2:**

**1. Is this Decidable or Undecidable?**

**"Does a given Turing Machine M answer yes to a given problem?" undecidable**

**2. What is the implication of this in terms of analyzing computer programs?**

**Hint:**

**Think of “a given TM” as “a given computer program”.**

**It is undecidable whether a given computer program will answer yes to a given problem.**

**==================================================**

**Summary - Design your own language [14 pts]**

**===================================================**

**You must design a new language.**

**Hints:**

**C++ variables: (l|u)(l|u|d)\* (DFA with only 2 states)**

**C++ couts: <outputstmt> ::= COUT << <expression> { << <expression> } SEMICOLON**

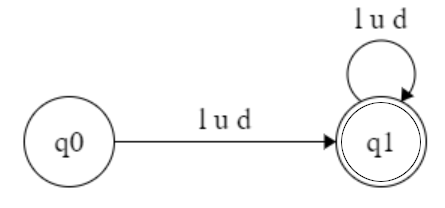
**Variable names in the new language can be of any length. You should allow more symbols than just letters, digits and underscore.**

**1) What should variable names look like?**

**a. Describe it as a regular expression (using \*, +, and/or |) [2]**

* **(l | u | d)+**

**b. And then draw its DFA (theory version is fine). [2]**

* ****

**Let’s assume that you did the same for other tokens in your language.**

**Now, you are designing statements in your language.**

**Output statements (like cout in C++) in the new language designed should allow any number of <expression>s to be output.**

**2) What should your output statement look like?**

**a. Give a context free grammar rule (in BNF) and you may refer to <expression>.[2]**

**<outputStmt> ::= PRINT ( <expression> {, <expression>} ) SEMICOLON**

**b. In what way is this better than cout of C++? [2]**

* **Print is a standard function call and does not use an operator like <<.**

**3) What proof technique can you use to prove that the grammar you have written is for the language you have designed? [2]**

* **Induction on the length of the strings in L.**

**4) What proof techniques (name 2) can you use to prove that the language another group designed is not Regular or is not Context Free? [2]**

* **Proof using pumping lemma**
* **Proof using closure**

**5) Finally, do you think we could have done the Project (scanner/parser) if the language was English? Explain based on your analysis of English. [2]**

* **No; how to parse “you read it” (imperative) vs “you read it” (past)? No syntactical difference**