“CS 374” Fall 2014 — Homework 2

Due Tuesday, September 16, 2014 at noon

Groups of up to three students may submit common solutions for each problem in this homework and in all future homeworks. You are responsible for forming you own groups; you are welcome to advertise for group members on Piazza. You need not use the same group for every homework, or even for every problem in a single homework. Please clearly print the names and NetIDs of each of your group members at the top of each submitted solution, along with ***one*** discussion section where we should return your graded work. If you submit hand-written solutions, please use the last three pages of this homework as templates.

---

1. ***C comments*** are the set of strings over alphabet $\Sigma = \{*, /, \texttt{A}, \diamond, \hookleftarrow\}$ that form a proper comment in the C program language and its descendants, like C++ and Java. Here $\hookleftarrow$ represents the newline character, $\diamond$ represents any other whitespace character (like the space and tab characters), and A represents any non-whitespace character other than $*$ or /.[1] There are two types of C comments:

   - Line comments: Strings of the form // ... $\hookleftarrow$.
   - Block comments: Strings of the form /* ... */.

   Following the C99 standard, we explicitly disallow ***nesting*** comments of the same type. A line comment starts with // and ends at the first $\hookleftarrow$ after the opening //. A block comment starts with /* and ends at the the first */ completely after the opening /*; in particular, every block comment has at least two *s. For example, the following strings are all valid C comments:

   - /***/
   - //◇//◇$\hookleftarrow$
   - /*///◇*◇$\hookleftarrow$**/
   - /*◇//◇$\hookleftarrow$◇*/

   On the other hand, the following strings are *not* valid C comments:

   - /*/
   - //◇//◇$\hookleftarrow$◇$\hookleftarrow$
   - /*◇/*◇*/◇*/

   (a) Describe a DFA that accepts the set of all C comments.

   (b) Describe a DFA that accepts the set of all strings composed entirely of blanks($\diamond$), newlines($\hookleftarrow$), and C comments.

   You must explain *in English* how your DFAs work. Drawings or formal descriptions without English explanations will receive no credit, even if they are correct.

2. Construct a DFA for the following language over alphabet $\{0,1\}$:

$$L = \left\{ w \in \{0,1\}^* \;\middle|\; \begin{array}{l}\text{the number represented by binary string } w \text{ is divisible} \\ \text{by 19, but the length of } w \text{ is not a multiple of 23}\end{array} \right\}.$$

You must explain *in English* how your DFA works. A formal description without an English explanation will receive no credit, even if it is correct. Don't even try to draw the DFA.

3. Prove that each of the following languages is *not* regular.

   (a) $\left\{ w \in \{0\}^* \;\middle|\; \text{length of } w \text{ is a perfect square; that is, } |w| = k^2 \text{ for some integer } k \right\}$.

   (b) $\left\{ w \in \{0,1\}^* \;\middle|\; \text{the number represented by } w \text{ as a binary string is a perfect square} \right\}$.

*4. **[Extra credit]** Suppose L is a regular language which guarantees to contain at least one palindrome. Prove that if an n-state DFA M accepts L, then L contains a palindrome of length polynomial in n. What is the polynomial bound you get?

[1] The actual C commenting syntax is considerably more complex than described here, because of character and string literals.

- The opening `/*` or `//` of a comment must not be inside a string literal (`"..."`) or a (multi-)character literal (`'...'`).
- The opening double-quote of a string literal must not be inside a character literal (`'"'`) or a comment.
- The closing double-quote of a string literal must not be escaped (`\"`)
- The opening single-quote of a character literal must not be inside a string literal (`"...'..."`) or a comment.
- The closing single-quote of a character literal must not be escaped (`\'`)
- A backslash escapes the next symbol if and only if it is not itself escaped (`\\`) or inside a comment.

For example, the string `"/*\\\"*/"/*"/*\"/*"*/` is a valid string literal (representing the 5-character string `/*\"\*/`, which is itself a valid block comment!) followed immediately by a valid block comment. For this homework question, just pretend that the characters `'`, `"`, and `\` don't exist.

The C++ commenting is even more complicated, thanks to the addition of *raw* string literals. Don't ask.

Some C and C++ compilers do support nested block comments, in violation of the language specification. A few other languages, like OCaml, explicitly allow nesting comments.

CS 374 Fall 2014 — Homework 2 Problem 1

Name:	NetID:
Name:	NetID:
Name:	NetID:
Section: 1 2 3	

---

1. Describe a DFA that accepts the set of all C comments.
2. Describe a DFA that accepts the set of all strings composed entirely of blanks(␣), newlines(↵), and C comments.

CS 374 Fall 2014 — Homework 2 Problem 2

Name:	NetID:
Name:	NetID:
Name:	NetID:
Section: 1 2 3	

Construct a DFA for the following language over alphabet $\{0, 1\}$:

$$L = \left\{ w \in \{0,1\}^* \;\middle|\; \begin{array}{l} \text{the number represented by binary string } w \text{ is divisible} \\ \text{by 19, but the length of } w \text{ is not a multiple of 23} \end{array} \right\}.$$

CS 374 Fall 2014 — Homework 2 Problem 3

Name:	NetID:
Name:	NetID:
Name:	NetID:
Section: 1 2 3	

Prove that each of the following languages is *not* regular.

1. $\{w \in \{0\}^* \mid$ length of w is a perfect square; that is, $|w| = k^2$ for some integer $k\}$.
2. $\{w \in \{0,1\}^* \mid$ the number represented by w as a binary string is a perfect square$\}$.

CS 374 Fall 2014 — Homework 2 Problem 4

Name:	NetID:
Name:	NetID:
Name:	NetID:
Section: 1 2 3	

Extra credit. Submit answers in the drop box for problem 1 (but don't staple problems 1 and 4 together.)

Suppose L is a regular language which guarantees to contain at least one palindrome. Prove that if an n-state DFA M accepts L, then L contains a palindrome of length polynomial in n. What is the polynomial bound you get?
