Study Guide for Midterm Exam 2

In this exam, you can use **one** cheat sheets of the standard US letter size.

The exam will cover the following topics and skills:

Recursion (Week 5) [~12pts]:

1) Convert an iterative algorithm into (naïve) recursive algorithm
2) Recognize iterative functions, non-tail recursive functions, and tail recursive functions
3) Explain why tail recursion is more efficient than naïve recursion
4) List methods to transform naïve recursion to tail recursion
5) Describe recursion tree for a function call.

Dynamic Programming (Week 7) [~8pts]:

1) Compare the divide-and-conquer method and dynamic programming
2) Describe the type of problems that dynamic programming is typically applied to.
3) What are steps to develop a dynamic programming algorithm.
4) What is the main technique used by dynamic programming? What are ways to implement dynamic programming?
5) Describe how dynamic programming was applied to the rod-cutting problem. Explain the recursive structures for the rod cutting problem.
6) Explain why the naïve recursive implementation for the rod-cutting problem is inefficient. Describe the recursive tree for a recursive call in that implementation.

Compilation Overview (Week 8) [14pts]:

1) List the phases of compilation in order from start to finish. Which phases belong to front end, which phases belong to back end? Which phases are optional? What tools are used in all phases?
2) Identify the input, output, and the tasks/purposes of a scanner. Identify *tokens* in a simple C program.
3) Identify the input, output, and the tasks/purposes of a parser. Explain structure of a *parse tree* in relation to *context-free grammars* rules.
4) Identify the input, output, and the tasks/purposes of a semantic analyzer. Explain what a (abstract) *syntax tree* is, and give examples of *semantic rules*.
5) Distinguish between static semantic and dynamic semantic. Give examples of dynamic semantics.
6) Identify the input, output, and the tasks/purposes of the “target code generation” phase.

Regular Expressions and Finite Automata (Week 9) [~20pts]:

1) Explain how regular expressions are defined. Recognize the meaning of regular expressions. What is ***regular set*** or ***regular language*** of a regular expression?
2) Define a deterministic finite automaton (DFA) formally.
3) Explain the computation of a DFA. What is the ***language*** of a DFA?
4) Identify two additional features of a nondeterministic finite automaton (NFA).
5) Explain the computation of an NFA.
6) Convert a regular expression into an equivalent NFA.
7) Explain the equivalence among regular expressions, DFAs, and NFA.

Racket: In this exam, you are expected to understand and apply the syntax of Racket for defining (naive) recursive functions**.**