Exam rules. The exam is open book and open notes—you can use any printed or handwritten material. However, no electronic devices are allowed. Anything with an on-off switch must be turned off. There is one exception: you can use a music player as long as nobody else can hear it and you just put it on shuffle and do not use its controls during the exam.

About the problems. These problems are similar to your homework problems. The same conventions about alphabets, etc. that were used on the homework will apply here.

Scoring. Ten problems, 10 points each, total 100 points.

1. Give a regular expression matching exactly those strings accepted by the following NFA:
Answer: 2
2. Convert the following NFA to an equivalent DFA:

Answer:
3. Indicate which of the following tasks can be done with a finite automaton (FA), and which can be done with a PDA but not with a FA. Label each choice with FA or PDA. In each example, “valid” just means, “has the right form for”, e.g. a valid U.S. phone number has a certain form involving digits, parentheses, and dashes, regardless of whether it actually connects to a working telephone; a valid credit card number satisfies certain relations between the digits but does not have to belong to an individual with good credit. Do not attempt to explain your answers.

(a) recognizing whether a string is a syntactically correct (compilable) Java program.
(b) recognizing whether a string is a valid SJSU course identifier (such as CS 154). Here “valid” means that the department exists, but not necessarily that the course number represents a real course; but the course number must be between 0 and 300.
(c) recognizing whether a string is a valid U.S. phone number.
(d) recognizing whether a string is a valid U.S. postal code (ZIP code); for example 95003 and 95003-4612 are both valid.
(e) recognizing whether a string is a valid social security number

Answer: (a) requires a PDA, but the other four can be done with an FA. The point is the same for all four: the legal strings have a fixed maximum length, so we can remember what has been seen so far in a state, and carry out any computation at all based on what has been seen so far. An alternative but less practical reason is that the sets of strings in question are all finite, and in principle every finite set is regular, but you wouldn’t want to check for validity that way!

4. Convert the following grammar to an equivalent PDA:

\[ S \rightarrow aSbb \mid SS \mid \epsilon \]

Answer: The start state is 0, the only accepting state is 1. The rules are as follows

\[
\begin{align*}
0, \epsilon, \epsilon \rightarrow 1, S \\
1, \epsilon, S \rightarrow \epsilon, aSbb \\
1, \epsilon, S \rightarrow \epsilon, SS \\
1, \epsilon, S \rightarrow \epsilon, \epsilon \\
1, a, a \rightarrow 1, \epsilon \\
1, b, b \rightarrow 1, \epsilon
\end{align*}
\]
5. A restaurant menu can be thought of as a single string, which contains some newline characters. It is composed of a restaurant name, followed by a newline, followed by sections, such as Appetizers, Soups, Salads, Entrees, Desserts, and Drinks. Each section has a title, followed by a newline and then a newline-separated list of items, ending in another newline. A section must have at least one item in it, and every menu must have at least one section. Each item is composed of a tab character, then the name of the item, optionally followed by more information in parentheses, another tab character, and a price. The price has to end in .95, but before the price can come one or two digits (not three-let’s not even think about such prices). For example, here is a short menu following these rules:

Luigi’s Chicago Restaurant
Appetizers
  Pakora (deep-fried vegetables in batter) 4.95
  Ginger sushi (with pink ginger) 6.95
Soups
  Minestrone 5.95

In this problem you are asked to give a grammar based on the rules given in English above, that generates legal menus. The item names and parenthetical information can be any non-empty strings composed of letters.

Answer: The start symbol is Menu. The rules are

\[
\begin{align*}
\text{Menu} & \rightarrow \text{RestaurantName Newline SectionList} \\
\text{SectionList} & \rightarrow \text{Section SectionList} \mid \epsilon \\
\text{Section} & \rightarrow \text{SectionTitle Newline ItemList} \\
\text{ItemList} & \rightarrow \text{Item Newline ItemList} \mid \text{Newline} \\
\text{Item} & \rightarrow \text{Tab ItemName Price} \mid \text{Tab ItemName (Info) Price} \\
\text{Price} & \rightarrow \text{Digit Digit .95} \mid \text{Digit .95} \\
\text{ItemName} & \rightarrow \text{String} \\
\text{Info} & \rightarrow \text{String} \\
\text{RestaurantName} & \rightarrow \text{String} \\
\text{SectionTitle} & \rightarrow \text{String} \\
\text{String} & \rightarrow \text{Letter} \mid \text{Letter String} \\
\text{Tab} & \rightarrow \text{‘\t’} \\
\text{Newline} & \rightarrow \text{‘\n’}
\end{align*}
\]
6. For a string of a’s and b’s, let \( f(\alpha) \) be the string \( \alpha \) with a changed to b and b changed to a. For example \( f(aab) = bba \). Please use the pumping lemma to prove that the set \( L \) of \( \alpha f(\alpha) \) such that \( \alpha \) is any string of a’s and b’s is not regular. For example, aabbba is in \( L \) because the last half, bba, is \( f \) of the first half.

**Common mistake to avoid:** Your proof will be (completely) wrong if it contains variables that are introduced out of thin air, such as mentioning a letter \( k \) without saying where it came from or what it means.

**Answer:** Assume, for proof by contradiction, that \( L \) is regular. Let \( N \) be given by the pumping lemma. Choose \( w = a^N b^N \). Then \( w \) is in \( L \). According to the pumping lemma we have \( w = xyz \) with \( |xy| \leq N \) and \( |y| \geq 1 \) and \( xy^j z \in L \) for \( j = 0,1,2,\ldots \). Since \( |xy| \leq N \) and the first \( N \) characters of \( w \) are a, we must have \( y = a^k \), and \( k > 0 \) since \( |y| \geq 1 \). Then \( xy^2 z = a^{N+k} b^N \), which must belong to \( L \) by the pumping lemma, but which does not belong to \( L \) since its first half contains no b but its last half does contain an a. That completes the proof.

7. Please write a Turing machine (program) that erases the middle character of its input, if the input is of odd length, and terminates without changing the input, if the input is of even length. For example, if started with aaaaabbbb, it terminates without changing the input, if started with aaaaabaaaa it terminates with aaaaaaaaa on the tape. Please comment your code so it won’t take me all day to figure it out. If you do not comment your code intelligibly, you will not get credit. Assume the input alphabet is \{a, b\}. Answer:

```
0,a->1,A,R // leave a marker to mark our place in the first half,
// and remember what was under the marker
0,b->1,B,R
1,a->1,a,R // move right to end or to marker
1,b->1,b,R
1,_->2,_,L // back up to last character
1,A->2,a,L // replace previous marker with original character
1,B->2,b,L
2,a->3,A,L // leave right marker
2,b->3,B,L
2,A->9,_,H // that's the middle character so erase it (and stop)
3,a->3,a,L // move left
3,b->3,b,L
3,A->0,a,R // replace marker with original character, move right
// and loop to the beginning
3,B->0,b,R
0,A->9,a,H // even length string, replace original character and halt
0,B->9,b,H
```
8. True or false:

(a) It’s impossible to write a Java program that takes Java programs $e$ as inputs, and produces the same output as program $e$ would produce. (Here the “output” of a Java program is what, if anything, it prints to the console.)

False. We can write a program in Java that simulates the operation of a Turing machine supplied as input. Indeed there is such a program (but in C#) in the textbook.

(b) It’s impossible to write a Java program that takes Turing machines $e$ as inputs, and prints out 1 if Turing machine $e$ halts (with no input) and 0 if it does not.

True. This is the “unsolvability of the halting problem.”

(c) It’s impossible to write a Java program that takes Java programs $e$ as inputs, and prints out 1 if program $e$ would eventually print something to the console, and 0 if not.

True. How long should we wait before deciding that it’s never going to print? Actually, we can prove the impossibility by reducing the halting problem to this set: Define, for programs $e$, the program $F(e)$ that first does what $e$ does, but suppressing any printout, and then, if and when $e$ halts, it prints “done”. Now $e$ halts if and only if $F(e)$ prints something. Hence the set in this question is not recursive.

(d) It’s impossible to write a Turing machine that would take Java programs $e$ as inputs, and produce the same output as program $e$ would produce.

False. We can write a Java interpreter in Turing machine language. There are some technicalities involved here but the principle is simply the Church-Turing thesis, that whatever can be computed can be computed on a Turing machine.

(e) It’s impossible to write a Java program that takes as input a Java program $e$ and a number $k$, and outputs 1 if program $e$ (without input) outputs $k$ (before printing anything else, although it may go on outputting other characters after printing the digits of $k$), and 0 otherwise.

True. Again, how long should we wait before deciding that $e$ is not going to output $k$? The argument given under part (c) can easily be adapted, replacing “done” by $k$. 

9. Please classify the following sets as either recursive or r.e. (recursively enumerable), as in the homework. You do not have to justify or explain your answers.

   (a) The set of doubled strings, i.e. those of the form $\alpha\alpha$ for some $\alpha$. *(Recursive)*

   (b) The set of strings rejected by a particular finite automaton. *(Recursive)*

   (c) The set of strings accepted by a particular Turing machine (assume here that I have listed a machine with several thousand instructions that I do not expect you to read and analyze). *(Recursively enumerable, and for some Turing machines, not recursive.)*

   (d) The set of strings of digits $\alpha$ that occur somewhere in the decimal expansion of $\pi$. For example, the string 4159 occurs, since $\pi$ starts out 3.1415923. *(Recursively enumerable. Whether the set is actually recursive or not is beyond current mathematical knowledge.)*

   (e) The set of Java programs that produce at least 1000 characters of output to the console. *(Recursively enumerable; this set is certainly not recursive, as discussed in the previous question.)*
10. Consider the *Partition Problem*. The input is a finite set of positive integers. The problem is to determine whether it can be partitioned into two disjoint sets $B$ and $C$ such that the sum of the elements in $B$ is equal to the sum of the elements in $C$. For example, if $A = \{1, 3, 5, 7\}$, we could take $B = \{3, 5\}$ and $C = \{1, 7\}$, but if $A = \{1, 3, 5, 78\}$, it is not possible to partition $A$ as required, since whichever set contains 78 will have too large a sum. Note, numbers are represented in ordinary decimal notation for purposes of this problem.

Is this problem in $\mathcal{P}$, or is it in $\mathcal{NP}$, or is it in $\text{Co-}\mathcal{NP}$? Briefly explain your answer.

*Answer:* It is in $\mathcal{NP}$, since we can have a non-deterministic Turing machine generate a subset $B$, then (deterministically and in polynomial time) compute the sum of the numbers in $B$, the sum of the numbers in $A$ but not in $B$, and see if they are equal.