198:205
Discrete Structures I
Professor McCarty

Sample Final Exam

Rules: This is a closed book exam, but you may have with you one (8 1/2 in. × 11 in.) sheet of paper with writing on both sides

Advice: Read each problem, think about it, read it again, and think about it again *before* you start writing. If you have a question, ask! Don't get bogged down on a particular problem. If you're stuck, go on to the next problem, and come back again later.

Question 1:

(10 points) Let A, B and C be arbitrary sets, and consider the set identities listed below:

(a) $A - (B \cap C) \quad ?=? \quad (A - B) \cap (A - C)$

(b) $A - (B \cap C) \quad ?=? \quad (A - B) \cup (A - C)$

One of these statements is true, and one is false. Which is which?

For the true statement, write out a proof, using a chain of logical equivalences or a chain of set identities. For the false statement, construct a counter example.

Question 2:

(20 points) Let $f : A \mapsto B$ be a function from a set A to a set B. We say that f *preserves proper subsets* if $f(A_1) \subset f(A_2)$ whenever $A_1 \subset A_2 \subseteq A$. (Recall that $A_1 \subset A_2$ if and only if $A_1 \subseteq A_2$ but $A_1 \neq A_2$.)

Prove the following:

(a) If f preserves proper subsets, then f is one-to-one. (Hint: Show the contrapositive!)

(b) If f is one-to-one, then f preserves proper subsets. (Hint: Show the contrapositive!)

Question 3:

(10 points) Use the weak principle of mathematical induction to prove that $2^{2n} - 1$ is divisible by 3 for every natural number n.

Question 4:

(25 points) Let `A` and `B` be two nondecreasing one-dimensional arrays of integers. Let `m` and `n` be the lengths, respectively, of `A` and `B`. (We are assuming in our pseudo code that array indices start with 1.) Finally, let `X` be a given integer.

We would like to write a procedure `findsum` that returns an ordered pair of indices `(i,j)` such that `A[i]` + `B[j]` = `X`, if such indices `i` and `j` exist. Otherwise, if there are no such indices, we would like `findsum` to return the ordered pair `(0,0)`. This is the purpose of the following code:

```
procedure findsum(A,B,m,n,X)
   i := 1
   j := n
   while i <= m and j >= 1
      begin
         if A[i] + B[j] = X then return (i,j)
            else if A[i] + B[j] < X
                 then i := i + 1
            else if A[i] + B[j] > X
                 then j := j - 1
      end
   return (0,0)
```

To test your understanding of this code, you might want to trace it on the following example:

A:	4	5	8	10	11	14	17		
B:	5	7	10	11	13	14	17	19	20

If you call `findsum(A,B,7,9,X)` on these arrays with `X = 20`, the procedure should return the value `(4,3)`, since `A[4] + B[3] = 20`. However, with `X = 26`, the procedure would find no match and should return `(0,0)`.

Can you prove that this code is correct? Here is a proposal for a loop invariant, P:

$$\exists u \exists v[\, 1 \le u \le \texttt{m} \;\wedge\; 1 \le v \le \texttt{n} \;\wedge\; \texttt{A}[u] + \texttt{B}[v] = \texttt{X} \,]$$

$$\leftrightarrow$$

$$\exists u \exists v[\, \texttt{i} \le u \le \texttt{m} \;\wedge\; 1 \le v \le \texttt{j} \;\wedge\; \texttt{A}[u] + \texttt{B}[v] = \texttt{X} \,]$$

Answer the following questions:

(a) Show that P is true when the execution of the procedure reaches the `while` loop.

(b) Show that P remains true whenever the execution of the `begin`/`end` block terminates normally. That is, verify the following partial correctness assertion:

$$(P \wedge C)\ \{S\}\ P$$

where S is the `begin`/`end` block and C is the condition: `i <= m and j >= 1`. (Note: There are two cases to consider here. What are they? If you analyze one case in detail, you need only write out an abbreviated analysis of the other case, which is similar.)

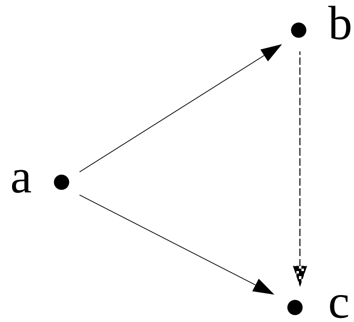
(c) Use the results of Parts (a) and (b) to show that `findsum` is correct. That is, show that the procedure exits the `while` loop normally and returns `(0,0)` if and only if there are no indices `i` and `j` such that `A[i]` + `B[j]` = `X`. It is convenient to split this biconditional into two parts:

(i) If `findsum` exits the `while` loop normally and returns `(0,0)`, then there are no indices `i` and `j` such that `A[i]` + `B[j]` = `X`.

(i) If there are no indices `i` and `j` such that `A[i]` + `B[j]` = `X`, then `findsum` exits the `while` loop normally and returns `(0,0)`. (Note: Be sure to include an argument that `findsum` terminates!)

Question 5:

(15 points) Recall that a relation R on a set A is an equivalence relation if it is reflexive, symmetric and transitive. Are there other ways to characterize equivalence relations? Consider the following:



> **Definition:** A relation R on a set A is *euclidean* if, for all $a, b, c \in A$: $(a, b) \in R$ and $(a, c) \in R$ implies $(b, c) \in R$.

The graph shown above may help you to visualize euclidean relations. Now prove the following:

> **Theorem:** If a relation R on a set A is both reflexive and euclidean, then it is an equivalence relation.

(Hint: You need to show that a reflexive and euclidean relation is also symmetric and transitive.)

Question 6:

(20 points) Let $T = \{(,)\}$, so that T^* is the set of all finite strings of left and right parentheses. We say that a string of parentheses, x, is *balanced* if: (1) x has the same number of left and right parentheses overall; and (2) in every initial segment of x, the number of left parentheses is greater than or equal to the number of right parentheses.

For example, consider the string: (()(()))()(). This string has a total of 6 left parentheses and 6 right parentheses, so the first part of the definition is satisfied. Furthermore, the initial segments of this string are: λ, (, ((, ((), (()(, (()((, (()((), (()(()), (()(())), (()(()))(, (()(()))(), (()(()))()(and (()(()))()(), and in each of these segments the number of left parentheses is greater than or equal to the number of right parentheses. Thus (()(()))()() is a balanced string.

Let $\mathbf{B} \subset T^*$ be the set of all balanced strings of parentheses. Let $\mathbf{G} = (V, T, S, P)$ be a grammar in which the start symbol, S, is the only nonterminal, and the productions are:

- $S \rightarrow (S) S$
- $S \rightarrow \lambda$

Prove the following:

(a) $\mathbf{B} \subseteq L(\mathbf{G})$. In other words, show that every balanced string of parentheses is generated by the grammar **G**. (Hint: Try a proof by strong induction on the length of a string.)

(b) $L(\mathbf{G}) \subseteq \mathbf{B}$. In other words, show that every terminal string generated by the grammar **G** is a balanced string of parentheses. (Hint: Try a proof by strong induction on the height of a derivation tree.)