CS312 Fall 2001
Prelim 2
November 20, 2001

Name_____

NetID_____**@cornell.edu**

Instructions

Write your name and netID on the lines above. There are seven questions on 15 numbered pages. Check now that you have all the pages. Write your answers in the boxes provided. Ambiguous answers will be considered incorrect. The exam is closed book except for the handout provided. Do not begin until instructed. You have 90 minutes. **Good luck!**

1	2	3	4	5	6	7	Σ
/14	/12	/9	/7	/9	/7	/3	/61

It was a dark and stormy night.

Suddenly, a shot did not ring out!

But don't worry, one will soon enough.

The assembled dinner guests have all arrived at the behest of a certain Mr. Boddy, proprietor of the mansion, who made a fortune in teaching CS312 to poor college students. None of the guests knew why they had been summoned, except through a vaguely worded invitation that they received in the mail weeks earlier:

CS312 Prelim 2
Issued 11/20, 7:30pm
Due 11/20, 9:00pm

Dear _____ :

You are cordially invited to a dinner prelim at Warren B45. It will be to your advantage to be present on this date because a Mr. Boddy will bring to an end a certain long-awaited academic liability.

Signed,
A friend

Colonel Mustard arrives last. "Ah, good sir, would you happen to be Mr. Boddy?"

"Oh, indeed, no, sir. I'm merely a humble butler."

"And what exactly do you do?"

"I buttle, sir."

The butler motions the guests towards the dining room. "You'll find your names beside your places. Please be seated."

"Colonel Mustard, Miss Scarlet, Mr. Green, Ms. White, Professor Plum, Mrs. Peacock, may I present to you Mr. Boddy..."

EVALUATE THE DINNER CONVERSATIONS, LISTENING CAREFULLY FOR CLUES

1. (14 points) Determine the value and type of each of the following expressions as it would be printed by the interpreter. Don't forget to include the type.

```
- val (s,t) = ([1,2,3],[4,5,6]);
val s = [1,2,3] : int list
val t = [4,5,6] : int list
```

(a) `- s@t;`
`val it =` **`[1,2,3,4,5,6] : int list`**

(b) `- s::[t];`
`val it =` **`[[1,2,3],[4,5,6]] : int list list`**

(c) `- map (fn z => z::t) s;`
`val it =` **`[[1,4,5,6],[2,4,5,6],[3,4,5,6]] : int list list`**

(d) `- map (fn z => map (fn w => w+z) s) t;`
`val it =` **`[[5,6,7],[6,7,8],[7,8,9]] : int list list`**

(e) `- let val s = tl t val t = hd s in t end;`
`val it =` **`5 : int`**

(f) `- let val x = ref 2 fun f y = x := !x + y in app f s end;`
`val it =` **`() : unit`**

(g) `- [[[]]];`
`val it =` **`[[[]]] : 'a list list list`**

"...And as you can see, Mr. Boddy knows of the skeletons in your closet, and has been using that information to blackmail all of you for the past few years."

"You bastard!" shouts Colonel Mustard. "Why, if I ever get my hands on..."

"And I could think of all sorts of unspeakable things to do to you..." says Miss Scarlet.

The Butler interrupts, and then continues. "Ladies and gentlemen, the police will be here in about an hour. Tell them the truth, and Mr. Boddy will be behind bars."

Mr. Boddy steps forward and presents each of the guests with some object. "If you denounce me to the police, you will also be exposed and humiliated. Given the stuff we've just heard, I'm sure none of you want that hitting the newspapers. But...if one of you kills the butler now...no one but the seven of us will ever know. I suggest you think about it." With that, Mr. Boddy walks past the butler, who is still standing with a look of shock on his face, and stands by the light switch.

The lights go out.

A gunshot is heard. The sounds of multiple blunt objects striking something is heard. A rope whipping through the air is heard, followed by several metal objects clanging on the ground.

The lights go on.

Mr. Boddy's body lies on the floor, dead.

HELP SOMEBODY KILL MR. BODDY BY DEFINING ZARDOZ IN DIFFERENT WAYS

2. (12 points) Supply a value for `zardoz` that causes the entire expression to evaluate to 312. If impossible, say so and explain why. Example:

```
let
  val zardoz = (312,313)
in
  #1 zardoz
end
```

(a)
```
let
  val zardoz = fn _ => 312
in
  zardoz (zardoz zardoz)
end
```

(b)
```
let
  val zardoz = fn _ => [312]
in
  hd (hd (map zardoz [zardoz]))
end
```

(c)
```
let
  val zardoz = fn _ => fn _ => fn _ => 312
in
  zardoz ()()()
end
```

(d)
```
let
  val zardoz = 3
  fun f n = 5*n + zardoz
in
  List.foldl (op * ) 1 (List.tabulate (zardoz,f))
end
```

"He's dead!"

"It's not the butler!" exclaims Colonel Mustard.

Miss White speaks. "Well, who had the gun? I heard a gunshot go off. So whoever had the gun must have killed Mr. Boddy."

Mrs. Peacock adds "But I thought I heard the sound of a blunt object hitting someone."

Miss Scarlet adds "And I heard a knife drop and clatter on the ground."

Mr. Green adds "I could swear I felt a rope brush by me."

At once, everyone starts talking and accusing everyone else.

Exasperated, the butler says "Look. Each of you has a deadly weapon in his or her hand. We heard sounds involving all of the weapons, and now Mr. Boddy is dead. One of us in this room must have killed Mr. Boddy."

Colonel Mustard speaks up. "I suggest we handle this in proper military fashion. We split up, and search the house..."

SPLIT THE CHARACTERS INTO GROUPS

3. (9 points)

(a) (8 points) Write a function `split` that, given an integer `i` and a list of elements `s`, splits the list into a triple consisting of the list of elements in `s` before the i^{th} element (starting at 0), the i^{th} element, and the list of elements in `s` after the i^{th} element. Raise `Subscript` if `i` is out of bounds. Examples:

```
- split 2 [5,6,7,8,9];
val it = ([5,6],7,[8,9]) : int list * int * int list
- split 0 ["purple","dinosaur"];
val it = ([],"purple",["dinosaur"]) : string list * string * string list
- split 3 [(),(),()];
uncaught exception subscript out of bounds
  raised at: stdIn:22.23-22.32
```

```
fun split i s =
  case (i,s) of
    (_,[]) => raise Subscript
  | (0,x::t) => ([],x,t)
  | (_,x::t) =>
      let val (u,y,v) = split (i-1) t
      in (x::u,y,v)
      end
```

(b) (1 point) What is the type of `split`?

```
int -> 'a list -> 'a list * 'a * 'a list
```

Somewhere along the line, the cook was also found dead. "The cooking wasn't that bad," they say. A singing telegram girl also shows up, and is promptly hastened off this mortal coil.

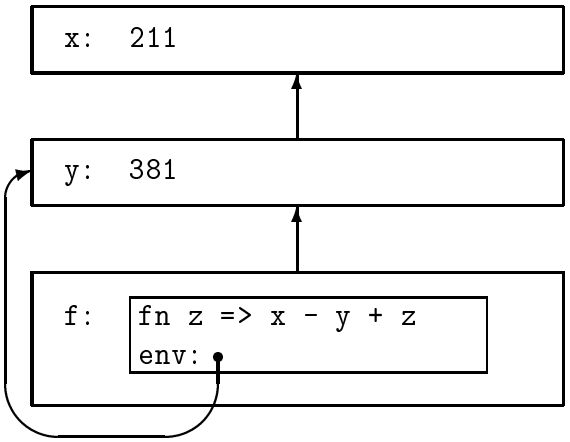
The maze of twisty passages, all alike, and the secret passageways all over the mansion, do not help things.

Mr. Green gets an idea. "Our problem is that we can't figure out who could have been where after the murder. We should look at the layout of the building and try to figure out who could have gone where..."

EXAMINE THE ENVIRONMENT OF THE MANSION TO FIGURE OUT WHO FRAMED THE GARBAGE COLLECTOR

4. (7 points)

(a) (3 points) Give a definition of `f` that results in the following environment.




```
val f = let
  val x = 211
  val y = 381
in
  fn z => x - y + z
end
```

(b) (3 points) Suppose we were to apply `f` to `482` in this environment. Draw the box-and-pointer diagram at the point immediately before the body of the function is evaluated.

The same as above, except with another binding `z : 482` pointing to the box `y : 381`. This is the environment in which `f 482` is evaluated.

(c) (1 point) What would be the value of `f 482`?

```
- f 482;
val it = 312 : int
```

"We know that the murders have to have been committed by someone here, somewhere in the mansion, using some of these weapons," says Mr. Green.

"Hmm..." mutters Professor Plum. Suddenly, he seems to get an insight (clue?): "Well, we know that some subset of the people here tonight, excluding the guests but including Mr. Boddy, have been murdered. We also know that some subset of the weapons, possibly non-strict, have been used to kill Mr. Boddy. Lastly, Mr. Boddy must have gotten killed various times in some subset of the rooms of this mansion, since we never figured out exactly when he died.

Miss Scarlet, ever practical, is the first to reply. "Um, brilliant, professor. But how exactly do we find this, er, subset of people, weapons, and places? What I want to know is: Who killed Mr. Boddy, in what room, with what weapon?"

"Ah, glad you asked," said the professor professorially. "Easy. We construct all the subsets of the three sets, and construct a sort of powerset containing all the combinations of people, weapons, and places we need to look at. At least, I think that works. Ordinarily I could find subsets easily recursively, but I'm running out of paper. Hmm... if only we could make it tail-recursive!

HELP THE PROFESSOR GENERATE SUBSETS IN CONSTANT STACK SPACE BY WRITING A TAIL-RECURSIVE FUNCTION

5. (9 points) Here is a program that computes all subsets of a set. The set is given by a list of distinct elements.

```
fun subsets (x:'a list) : 'a list list =
  case x of
    [] => [[]]
  | x::t =>
      let val st = subsets t
      in st @ (map (fn z => x::z) st)
      end

- subsets ([]:int list);
val it = [[]] : int list list
- subsets [1];
val it = [[],[1]] : int list list
- subsets [1,2];
val it = [[],[2],[1],[1,2]] : int list list
- subsets [1,2,3];
val it = [[],[3],[2],[2,3],[1],[1,3],[1,2],[1,2,3]] : int list list
```

Rewrite this program in tail recursive form. The order of the subsets or the elements in each subset does not matter—your program may produce them in a different order than above if more convenient. You may assume that all the elements of the input list are distinct.

```
fun subsets (x:'a list) : 'a list list = let
  fun subsets' (y:'a list) (out:'a list list) : 'a list list =
    case y of
      [] => out
    | u::t => subsets' t (out@(map (fn z => u::z) out))
in
  subsets' x [[]]
end
```

(Note: the type of `out` was erroneously given as `'a list` in the problem.)

Nobody is any closer to figuring out whodunit.

"We can work through this logically," said the butler. "Since we assume people were getting killed because they know something, then we should be able to figure out who knew what by who killed whom."

Miss Scarlet says "All right, genius, let's hear what you have to say." She reclines back on the couch and sips her drink.

The butler clears his throat. "Let's start. We know whoever killed Mr. Boddy would also need to kill whoever knew he or she was going to kill Mr. Boddy. But all the dead people knew at least one other. So if you killed Mr. Boddy and whoever knew you would kill Mr. Boddy, you would have to kill at least one other person, because that person would have figured out from the death of the second person who killed Mr. Boddy. But then a third person would observe the death of the second other person, and from that, deduce who must have killed the first other person, and therefore whoever killed Mr. Boddy."

Mrs. Peacock gets a bright insight briefly. "So, does that mean that by induction, if you killed Mr. Boddy, you'd have to kill a lot of other people too?"

PROVE BY INDUCTION WHAT YOU'VE ALREADY LEARNED IN THE PREVIOUS PROBLEM

6. (7 points) Prove by induction on n that the program given to you in Problem 5 (not the one you have written!) produces a list of length 2^n when given an input list of length n (by convention, $2^0 = 1$). You may use without proof the facts that `@` produces a list whose length is the sum of the lengths of its two arguments and that `map` produces a list of the same length as its second argument.

Basis: $n = 0$. In this case the input list is `[]`. The first clause of the case expression is evaluated, giving `[[]]`, which is of length $1 = 2^0$, as desired. This establishes the basis.

Induction step: Assuming that the statement holds for n, we show that it holds for $n + 1$. An input list of length $n + 1$ must be of the form `x::t`, where `t` is of length n. In this case the second clause of the case expression is evaluated. This recursively computes `st = subsets t`, which by the induction hypothesis is of length 2^n. By the fact about map given in the problem statement, the value `(map (fn z => x::z) st)` is of length 2^n as well. Also, by the fact about `@` given in the problem statement, the value `st @ (map (fn z => x::z) st)` is of length $2^n + 2^n = 2^{n+1}$. This is the final result of the computation, so the length of the output list is 2^{n+1}, as was to be shown.

The Butler continues: "Very well... I know who did it." Incredulous gasps arise from all the guests. "And I'm going to show you how it was done. Follow me." The guests all file into the library, arguing amongst themselves.

Briefly, the butler, with the help of everyone still alive, sketches the events of the evening, and the relevant facts. He continues with the chain of logical derivations.

"... The wounds on the cook and the singing telegram girl match, so assuming the murderer kept the same weapon, whoever killed the cook must also have killed the singing telegram girl."

"We also know that whoever found the second body first must have been with someone else in the lounge at that time the secret passage was opened, in order for someone to slip out the secret passage to the conservatory. But when this happened, someone must have come by and moved Mr. Boddy."

"The person who moved Mr. Boddy cannot be anyone who was in the lounge at the time, and couldn't have entered from the hall because the door was locked shut. Therefore, that person must have come in from the conservatory. But no one who did the murder was in the conservatory at the time, so that person could not have murdered Mr. Boddy, but merely accidentally helped whoever did. Is everything clear?"

Colonel Mustard exclaims "Ah! I get it! Then whoever was in the study at the time whoever killed the cook also killed the singing telegram girl must have been in the lounge with whoever was with the person who found the second body first when the person who accidentally opened the secret passage let whoever was missing from the conservatory slip out and foil whoever was trying to cover the tracks of whoever killed Mr. Boddy!"

"Yes, it makes perfect sense now" says Miss Scarlet. "I think we all know who did it. Kind of surprising, really, that it should be..."

Everyone turns to face the guilty party.

SOLVE THE MYSTERY BY UNIFYING THE MURDER EQUATIONS

7. (3 points) Let `f,g,a,b` be function and constant symbols and `X,Y,Z` variables. Say whether the following pairs of terms unify, and if so, give their most general unifier. Example:

	`f(X,a)`	`f(b,Y)`	`X←b, Y←a`

(a)	`f(X,g(X,Y))`	`g(X,f(X,Y))`	`not unifiable`
(b)	`f(g(X,Y),g(Y,X))`	`f(g(a,a),Z)`	`X←a, Y←a, Z←g(a,a)`
(c)	`f(X,g(a,X))`	`f(g(b,a),X)`	`not unifiable`

Extra credit ($\frac{1}{2}$ point): Who killed Mr. Boddy? `Mr. Boddy himself`