Question 8 (4 points):

Note: This is a hard question! We expect that many people won't solve it.

Think about a predicate function of several Boolean arguments, made up of some composition of `and`, `or`, and `not`, like these examples:

```
(define (fun p q r s t)
  (and (or p q)
       (not (or q r s))
       (and s t)))
```

```
(define (zot p q r s t)
  (and (or p q)
       (not (or q r s))
       (and p t)))
```

Such a predicate is called *satisfiable* if there is some combination of argument values that will make this function return `#T`. For example, the function `fun` is *not* satisfiable, because the argument called `s` would have to be false to satisfy `(not (or q r s))` but would have to be true to satisfy `(and s t)`, so those subexpressions can't both be true. But `zot` *is* satisfiable, because `(zot #t #f #f #f #t)` returns `#T`.

Write a function `satisfiable?` whose two arguments are a predicate function such as the example described above and a number indicating how many arguments that function takes. Your function should return true if the argument function is satisfiable. For example:

```
> (satisfiable? fun 5)
#F
```

Hint: Create all possible combinations of Boolean argument values.

Your name ____ login cs3–_______

Question 7 (4 points):

You've seen in the spreadsheet example that vectors can have vectors as elements. Those sub-vectors might themselves have sub-sub-vectors as elements, and so on. Write a predicate `vector-deep-member?` that takes two arguments, the first of which is any value and the second of which is a vector, possibly including subvectors. The predicate should return true if its first argument is a member of the vector, or a member of a member, etc.

```
> (vector-deep-member? 3 #(4 #(27 92) #(3 #(7 15 4) 9) 8 #(2 5)))
#T
```

Question 6 (4 points):

Write a function `max-children` that takes a tree as its argument and returns the largest number of children that any node in the tree has. Use the tree abstraction, not "cheap" trees. For example, if `world-tree` is the tree illustrated on page 298 of the text, then

```
> (max-children world-tree)
7
```

because the node with the largest number of children in this example is the root node, with seven children.

Your name ______________________________ login cs3–______

Question 5 (4 points):

Write a predicate function `samepairs?` that takes a list as its argument. It should return true if the same two values appear twice as *consecutive* elements of the list. For example:

```
> (samepairs? '(a b x y c d e f x y g h))
#T
> (samepairs? '(a b x y c d e f x g y h))
#F
> (samepairs? '(a b x x x c d))
#T
```

(In the third example, there are two overlapping `x x` pairs.)

Question 4 (4 points):

Write a procedure `pairs-checker` whose argument is a predicate function of two arguments. It should return a predicate function of one argument, a list, that returns true if the original predicate is true for every pair of consecutive elements in the list.

For example:

```
> (define increasing? (pairs-checker <))
> (increasing? '(4 23 72 95 100))
#T
> (increasing? '(4 23 95 72 100))
#F

> ((pairs-checker equal?) '(foo foo foo foo foo))
#T
> ((pairs-checker equal?) '(foo foo foo foo baz baz))
#F
```

Your name ______________________________________ login cs3–_______

Question 2 (3 points):

Which of the following might be a sensible procedure argument to `repeated` with a numeric argument of 4? That is, for which of them will `repeated` return a function that can be applied to some argument without causing an error? (There may be more than one such procedure; circle all of them.)

`cdr`	`(lambda (x) (* x x))`
`cons`	`random`
`even?`	`length`

Question 3 (4 points):

Write a procedure `middle` that finds the middle word of a sentence. If the sentence has an even number of words, return the first of the two middle ones. For example:

```
> (middle '(got to get you into my life))
YOU

> (middle '(your mother should know))
MOTHER
```

Question 1 (3 points):

What is the value of each of the following expressions? (If the expression's value is a procedure you may just write "procedure." If evaluating the expression would cause an error, just write "error.")

```
> (caddar '((a b c d e f) (g h i j k l) (m n o p q r)))
```

```
> (append (list '(a b) '(c d)) (cons '(e f) '(g h)))
```

```
> (every (lambda (x) (item x '(a b c d e f g h))) 2514)
```

```
> (keep (lambda (x) (even? (count x))) '(john paul george ringo))
```

```
> (leaf? (make-node '(a b) '()))
```

```
> (filter (lambda (x) #t) '(back in the ussr))
```

CS 3 Final — May 16, 1994

Your name ____

login cs3–_______

This exam is worth 30 points, or about 28% of your total course grade. The exam contains eight questions.

This booklet contains eight numbered pages including the cover page. Put all answers on these pages, please; don't hand in stray pieces of paper. This is an open book exam.

When writing functions, write straightforward code. Do not try to make your program slightly more efficient at the cost of making it impossible to read and understand.

When writing procedures, don't put in error checks. Assume that you will be given arguments of the correct type.

Our expectation is that many of you will not complete one or two of these questions. If you find one question especially difficult, leave it for later; start with the ones you find easier.

1	/3
2	/3
3	/4
4	/4
5	/4
6	/4
7	/4
8	/4
total	/30