Name:____

CSE 303, Spring 2005, Midquarter Examination 29 April 2005

Please do not turn the page until everyone is ready.

Rules:

- The exam is closed-book, closed-note, except for one side of one 8.5x11in piece of paper.
- **Please stop promptly at 3:20.**
- You can rip apart the pages, but please write your name on each page.
- There are **85 points** total, distributed **unevenly** among 8 questions (some of which have multiple parts).
- When writing code, style matters, but don't worry about indentation.

Advice:

- Read questions carefully. Understand a question before you start writing.
- Write down thoughts and intermediate steps so you can get partial credit.
- The questions are not necessarily in order of difficulty. **Skip around.**
- If you have questions, ask.
- Relax. You are here to learn.

Name:____

1. (7 points) What do each of the following csh commands do:

   (a) `rm x`

   (b) `rm *x`

   (c) `rm * x`

   (d) `rm "* x"`

   Solution:

   (a) Deletes the file named x in the current directory.

   (b) Deletes all files in the current directory with names ending in x (and not starting with .).

   (c) Deletes all files (not starting with .) in the current directory.

   (d) Deletes the file in the current directory with the three-character name * x.

Name:____

2. (15 points) For each of the following, give a regular expression suitable for `grep` that matches the lines described:

   (a) Lines containing the string `dan`.

   (b) Lines containing the string `dan` but with any number of the characters capitalized.

   (c) Lines containing a `d` and an `a` and an `n` in that order, but with any number of characters in-between.

   (d) Lines containing the (five-character) string `[dan]`.

   (e) Lines *not* containing any lower-case English vowels.

Solution:

(a) `dan`

(b) `[dD][aA][nN]`

(c) `d.*a.*n`

(d) `\[dan\]`

(e) `^[^aeiou]*$`

Name:____

3. (10 points) Write a (very, very short) *shell script* that is like `dos2unix` except it stores the result in a different file.

   - Include the necessary first line for a shell script.
   - Take 2 arguments: the first is the input file; the second is the output file.
   - Do not include error-handling (assume the input file exists).
   - A DOS file has each line end with two characters: `\r` and then `\n`. A UNIX file has each line end with one character `\n`.

     So your script should remove (replace with the empty string) every `\r` that is at the end of a line.

   Note: If a file already has UNIX-style line endings, your script should be equivalent to copying the file.

   Solution:

```
#!/bin/csh
sed 's/\r$//' $1 > $2
```

Name:____

4. (5 points) Suppose you find the following line in a csh script:

```
echo `someProgram`  # these are backquote characters
```

Describe why this line is bad style. Give a simpler way of doing the same thing.

Solution:
This lines takes the standard-out from executing someProgram, makes it a string inside the csh script and then uses echo to send the string to standard-out. It is equivalent just to write someProgram.

Name:____

5. (13 points) Write a C function `increasing` that takes a pointer to a `struct MidtermList` and returns 1 if the linked list represented by the argument has *increasing* elements (each number less than the one after it) and 0 otherwise. As usual, `NULL` is the empty list.

   Big hint: All lists with 0 or 1 elements are "increasing". A longer list is increasing if its first element is less than its second and the "list without the first element" is increasing. Solutions "remembering the previous element" are possible but complicated — look at the "next" element instead.

```
struct MidtermList {
  int num;
  struct MidtermList * rest;
};
```

Solution:

```
int increasing(struct MidtermList * lst) {
  if(lst==NULL)
    return 1;
  for(; lst->rest != NULL; lst = lst->rest)
    if(lst->num >= lst->rest->num)
      return 0;
  return 1;
}
// or a recursive one (consumes more space in C, but easier to get right)
int increasing(struct MidtermList * lst) {
  if(lst == NULL || lst->rest == NULL)
    return 1;
  if(lst->num >= lst->rest->num)
    return 0;
  return increasing(lst->rest);
}
// or without multiple returns
int increasing(struct MidtermList * lst) {
  return lst == NULL || lst->rest == NULL
         || (lst->num < lst->rest->num && increasing(lst->rest));
}
```

Name:____

6. (15 points) Indicate the values for `a` and `b` using the blanks at each program point indicated below. Warning: This problem is tricky.

```
#include <stdio.h>

#define FOO(x,y) { printf(" %d ",y); x = y; }

void foo(int x, int y) { printf(" %d ",y); x = y; }

int main(int argc, char**argv) {
  int a = 2;
  int b = 3;

  FOO(a,++b);

   // a is ______________   b is ______________

  FOO(a,++b);

   // a is ______________   b is ______________

  foo(a,++b);

   // a is ______________   b is ______________

  foo(a,++b);

   // a is ______________   b is ______________

  return 0;
}
```

Solution:
5 5
7 7
7 8
7 9

Name:____

7. (15 points) Consider this C function:

```
void f(int * x, int * y) {
  free(x); // line 1
  *y = 17; // line 2
}
```

Describe 4 ways (not including the example below) that a caller could use `f` such that line 1 is *perfectly legal* but line 2 *might set the computer on fire.* Exactly 1 of your 4 ways must involve `x`.

Example: The caller could pass `NULL` for `y`.

Solution:

(a) The caller could pass an uninitialized value for `y`.

(b) The caller could pass a dangling pointer for `y`.

(c) The caller could cast an `int` to a pointer and pass that for `y`.

(d) The caller could pass the same pointer for `x` and `y` (making the assignment a dangling-pointer dereference).

Name:____

8. (5 points) Fill in the blanks so the program below prints `64`.

____

____

____

```
int f(int (*h)(int)) {
  return (*h)(13);
}
int main(int argc, char**argv) {
   printf("%d",f(&g));
}
```

Solution:
(Many other solutions exist.)

```
int g(int x) { return 64; }
```