Name:_____

CSE 303, Spring 2006, Midterm Examination
1 May 2006

Please do not turn the page until everyone is ready.

Rules:

- The exam is closed-book, closed-note, except for one 8.5x11in piece of paper (both sides).
- **Please stop promptly at 3:20.**
- You can rip apart the pages, but please write your name on each page.
- There are **75 points** total, distributed **unevenly** among 6 questions (all of which have multiple parts).

Question	Max	Grade
1	12	
2	9	
3	4	
4	15	
5	25	
6	10	
Total	75	

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. (**12** points)

   What does each one of the following commands do?

   (a) `ls ~/hw2/ >> documents`

   (b) `cat my-file | less`

   (c) `mv mytaxes*200[0-4]* old-tax-documents/`

   (d) `wc -l *.html | sort -n | head -n 1`

   Assume that `wc -l *.html` by itself produces the following output:

```
  624 course_dictionary.html
   25 grades.html
   28 index.html
   65 mainpage.html
  166 overview.html
   31 resources.html
  204 schedule.html
 1745 total
```

   Solution:

   (a) Appends the content of the subdirectory `hw2` in the user's home directory to a file called `documents`.

   (b) Pipes the content of the file `my-file` into `less`, which enables viewing the content by scrolling and searching.

   (c) Moves all documents that start with the string `mytaxes` and contain the string `2000`, `2001`, `2002`, `2003`, or `2004` to the directory `old-tax-documents`.

   (d) Finds the file with extension `.html` that has the fewest lines. Outputs the number of lines in that file and the name of that file. In the example, the output will be: `25 grades.html`.

Name:_____

2. (**9** points) For each of the following, give a regular expression suitable for `grep` (or `egrep`) that matches the lines described:

   (a) Lines that contain the letters `a`, `b`, and `c`, in that order, with at most one character between consecutive letters. For example, `abc` would match, so would `axbxc`, but `axxbxxc` would not.

   (b) Lines that start with the word `dog` and end with the word `cat`.

   (c) Lines that contain a floating point number. The number must have a decimal point and at least one digit before and one digit after the decimal point.

   Solution:

   (a) `a.?b.?c`

   (b) `^dog.*cat$`

   (c) `[0-9]+\.[0-9]+`

Name:_____

3. (**4** points) Explain what the following `sed` command does. Read the command carefully.

```
sed -e 's/dog/cat/' -e 's/puppie/kitten/g' animals.txt
```

Solution:
For each line in file `animals.txt`, replaces the first instance of `dog` with `cat` and all instances of `puppie` with `kitten`. Writes the modified file to `stdout`. Does not modify the original file.

Name:_____

4. (**15** points) Explain the behavior of the bash script below. Do not explain *how* it works, just the effects that a user of the script will see.

Reminders:

- `$@` refers to the list of command line arguments starting with `$1`.
- `date` prints the current date.
- `[ -d name ]` tests if a file `name` is a directory.

```
#!/bin/bash

if [ $# -lt 2 ]
then
    echo "usage: ‘basename $0‘ output_file dir1 dir2 ... dirn" >&2
    exit 1
fi

output_file=$1
echo "Snapshot as of ‘date‘" > $output_file

for arg in $@
do

    if [ -d $arg ]
    then

        for i in ${arg}/*.{c,h}
        do
            new="${i}.back"
            echo "$i $new" >> $output_file
            cp $i $new
        done

    fi

done
```

Solution:
If the number of arguments is less than two, outputs the usage to `stderr` (including the name of the script (without path)), and exits with error code 1.

Otherwise, uses the first command line argument as the log file and the remaining arguments as the list of directories to backup.

Writes a message to the log file that includes the current date.

Iterates over all directories, making a backup copy of all files with extension `.c` or `.h`. The backed-up files have the same name as the original files plus the extension `.back`. Appends the original and new name of each affected file to the log file.

Name:_____

[Problem 4 continued]

Name:_____

5. (**25** points)

   (a) Indicate the output of the program using the provided blanks.

```
#include <stdio.h>

#define SIZE 100

int main() {

    int array[SIZE];
    int i;
    for ( i = 0; i < SIZE; i++) {
        array[i] = i;
    }

    int *p1;
    int *p2;

    p1 = &array[1];
    p2 = array + 4;
    printf("%d %d\n", *p1, *p2);    // Output: _____________

    p2 = p1;
    *p2 = 25;
    printf("%d %d\n", *p1, *p2);    // Output: _____________

    int **pp1;
    int **pp2;

    pp1 = &p1;
    pp2 = &p2;

    *pp2 = p1 + 4;
    *(*pp1) = 50;
    printf("%d %d\n", *p1, *p2);    // Output: _____________

    return 0;
}
```

Solution:

```
1 4
25 25
50 5
```

Name:_____

[Problem 5a continued]

Name:_____

(b) Next to each sequence of instructions, indicate if it is correct or if there is a bug. If there is a bug, say what the problem is:

```
// Sequence 1
    int *p1 = (int *)malloc(sizeof(int));
    int *p2 = p1;
    *p1 = 10;
    free(p2);
    p1 = NULL;
    p2 = NULL;
```

```
// Sequence 2
    int *p1 = (int *)malloc(sizeof(int));
    int *p2 = p1;
    *p1 = 10;
    free(p1);
    free(p2);
    p1 = NULL;
    p2 = NULL;
```

```
// Sequence 3
    int *p1 = (int *)malloc(sizeof(int));
    int *p2 = (int *)malloc(sizeof(int));
    *p1 = 10;
    free(p1);
    free(p2);
    p1 = NULL;
    p2 = NULL;
```

```
// Sequence 4
    int *p1 = (int *)malloc(sizeof(int));
    int *p2 = (int *)malloc(sizeof(int));
    p1 = p2;
    free(p1);
    p1 = NULL;
    p2 = NULL;
```

Solution:

- Sequence 1: ok
- Sequence 2: deallocating the same chunk of memory twice.
- Sequence 3: ok
- Sequence 4: memory leak, will never be able to free the memory allocated for the first integer.

Name:_____

[Problem 5b continued]

Name:_____

6. (**10** points) Indicate the output of the program using the provided blanks. Warning: be careful!

```
#include <stdio.h>

#define INCR 2

#define PRINT(x) printf("%d\n", x )

#define COMPUTE(x)  (x * x + INCR)

int compute(int x ) {

    return ( x * x + INCR );

}


int main() {

  int a = 2;
  int b = 3;

  PRINT( compute(a) );    // Output is: ___________

  PRINT( COMPUTE(a) );    // Output is: ___________

  PRINT( compute(a+b) );  // Output is: ___________

  PRINT( COMPUTE(a+b) );  // Output is: ___________

  return 0;

}
```

Solution:

```
6
6
27
13
```