CS 451/551 Midterm

Date: Mar 23, 2016 **Max Points**: 100

Open book, open notes. No electronic devices

90 Minutes

Please justify all your answers.

Important Reminder As per the course Academic Honesty Statement, cheating of any kind will minimally result in receiving an F letter grade for the entire course.

1. You are given a file of fixed-size records of size `REC_SIZE`. These records can be compared using the following function:

```
/** Returns < 0, 0, >0 depending on whether the record
 *  pointed to by recP1 is less than, equal, greater than
 *  the record pointed to by recP2.
 */
int recCmp(const void *recP1, const void *recP2);
```

   The quick sort function from `stdlib.h` has the following prototype:

```
/** Sort array of nElements elements of size elementSize
 *  in ascending order (ordered by comparison function pointed
 *  to by compar).  The comparison function returns < 0, 0, > 0
 *  depending on whether the element pointed to by its
 *  first argument is less than, equal, greater than the
 *  element pointed to by its second argument.
 */
void qsort(void *base, size_t nElements, size_t elementSize,
           int (*compar)(const void *, const void *));
```

   Describe how you would use `qsort()` to sort the specified file of records in **non-ascending** order assuming that:

   a) No more than two records may be in memory at a time.

   b) The amount of memory used by the program may be proportional to the number of records in the file (but less than the size of the file).

   It is sufficient to provide a detailed description, actual code is not required. *20-points*

2. Given the following declaration:

```
int *(*f)(void (*)(), int (*)(int *));
```

   describe the type of `f` precisely. *15-points*

3. User u1 belongs to primary group g1 and supplementary group g2, and user u2 belongs to primary group g2 and supplementary group g1. Neither u1 nor u2 are super-users, nor do they belong to any supplementary groups other than those explicitly listed above. Given the following `ls -l` listing:

```
-rwxr-xr--    1 u1       g3          14096 Mar  8 22:38 exec1
-rwsr--r-x    1 u2       g2          44096 Mar 10 01:36 exec2
-r--r--rw-    1 u1       g1           4012 Mar 10 01:12 data1
-rw--w-r--    1 u2       g2           8222 Mar  8 17:13 data2
```

fill in the following matrix with a Y, N or - depending on whether the execution specified by the row can (Y) or cannot (N) make the access specified by the column (R denotes *read*, W denotes *write*), or such access does not make sense (-). Please remember to justify your answers. *15-points*

	data1 R	**data1 W**	**data2 R**	**data2 W**
u1 runs exec1				
u2 runs exec1				
u1 runs exec2				
u2 runs exec2				

4. List bugs and inadequacies in the following program which purports to print out the average of all the doubles read from the binary files specified by its command-line arguments.

```
int main(char *argv[], int argc) {
  double sum = 0.0;
  int n = 0;
  for (int i = 0; i < argc; i++) {
    FILE *f = fopen(argv[i], "r");
    double d;
    while (!feof(f)) {
      if (fread(d, sizeof(double), 1, f) != 1) {
        perror("fread"); exit(0);
      }
      sum += d; n++;
    } //while
  } //for
  printf("%g\n", sum/n);
} //main
```

You may assume that all required header files have been included. *15-points*

5. Write a function with the following specification:

```
/** Fill in the 10-character string pointed to by perms[] with
 *  the ‘ls -l‘ style permissions string for the file
 *  specified by file descriptor fd.
 *
 *  Specifically, perms[0] should be set to ’d’ if fd
 *  specifies a directory, ’l’ if fd specifies a symlink, ’b’
 *  if fd specifies a block special file, ’c’ if fd specifies
 *  a character special file, ’p’ if fd specifies a FIFO, ’s’
 *  if fd specifies a socket, ’-’ if fd specifies a ordinary
 *  file.
 *
 *  perms[1] ([4], [7]) should be set to ’r’ if the file
 *  specified by fd is readable by its owner (group, other),
 *  ’-’ otherwise; perms[2] ([5], [8]) should be set to ’w’ if
 *  the file specified by fd is writable by its owner (group,
 *  other), ’-’ otherwise; perms[3] ([6]) should be set to ’s’
 *  if the file specified by fd is executable by its owner
 *  (group) and the setuid-bit (setgid-bit) is set, as ’x’ if
 *  the file specified by fd is executable by its owner
 *  (group) but the setuid (setgid) bit is not set, as ’-’
 *  otherwise; perms[9] should be set to ’x’ if the file
 *  specified by fd is executable by other, ’-’ otherwise.
 *  The function should NUL-terminate the perms[] string.
 *
 *  The function should return 0 if there is no error;
 *  otherwise it should return the ‘errno‘ of the first error
 *  encountered.
 */
int getFilePermissions(int fd, char perms[11]);
```

You need not show the inclusion of required header files. *20-points*

6. Discuss the validity of the following statements: *15-points*

   a) If a setuid program is executed, then a necessary condition for the executing program to read a file is that the owner of the program have read access to the file.

   b) The output of `printf()` will always be line-buffered.

   c) A expression like `scanf("%d", i)` is always incorrect.

   d) If you do a `fprintf()` to a `FILE` stream, followed subsequently by a `write()` to the file descriptor underlying the same stream, then it is possible that the output of the `fprintf()` appears after the output corresponding to the `write()`.

   e) If a successful `close(0)` is immediately followed by a successful call to `open()`, then the return value of `open()` is guaranteed to be `0`.