CS 246
C and Unix
Exam 2

Name	

Instructions: For complete programs include any required header files. You don't need to write any comments.

Question	Points	Bonus Points	Score
1	36	0	
2	6	0	
3	8	0	
4	10	0	
5	10	0	
6	10	0	
7	10	0	
8	10	0	
9	0	10	
Total:	100	10	

1. (36 points) Write a SINGLE regular expression that can be used with egrep with NO options to match each set of lines as described below. Do not give multiple expressions. Make your expression as simple as possible. You don't need to write quotes.

   (a) Lines containing the string alpha.
   `alpha`

   (b) Lines beginning with alpha.
   `^alpha`

   (c) Lines ending with omega.
   `omeage$`

   (d) Lines beginning with alpha and ending with omega.
   `^alpha.*omega$`

   (e) Lines containing alpha or omega.
   `alpha|omega`

   (f) Lines containing alpha followed by omega (possibly with characters in between).
   `alpha.*omega`

   (g) Lines beginning with alpha that have at least 8 characters.
   `^alpha...`

   (h) Lines ending with omega that have at most 8 characters.
   `...omega$`

   (i) Lines containing an integer between 1000 and 9999.
   `(^|[^0-9])[1-9][0-9]{3}([^0-9]|$)`

   (j) Lines that are empty.
   `^$`

   (k) Lines that contain no digits (0-9).
   `^[^0-9]$`

   (l) Lines that are empty or consist entirely of digits.
   `^[0-9]*$`

2. (6 points) Give the octal permissions that result in each symbolic permission.

   (a) `r---w--wx`
   423

   (b) `rw-r-x-w-`
   652

3. (8 points) Circle True or False for each statement.

   (a) True [False] You can make a hard link to a directory.

   (b) [True] False There can be more than one entry in a directory for the same file.

   (c) True [False] A directory contains the permissions for a file.

   (d) [True] False An inode contains the last time a file was modified.

4. (10 points) Write a complete C program that reads characters from standard input (until end of file) and writes them to standard output, changing all newlines to spaces.

```
#include <stdio.h>

int main() {
  int c;

  while ((c = getchar()) != EOF) {
    if (c == '\n') {
      putchar(' ');
    }
    else {
      putchar(c);
    }
  }
}
```

5. (10 points) Write a complete C program that reads integers from standard input (until end of file) and prints their average.

```
#include <stdio.h>

int main() {
  int n;
  int sum = 0;
  int count = 0;

  while (scanf("%d", &n) == 1) {
    sum += n;
    count++;
  }
  printf("%f\n", sum / (double) count);
}
```

6. (10 points) Write a C program that takes any number of command line arguments (all integers) and prints their sum. If no arguments are present it will print 0. Do not do any error checking.

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
  int sum = 0;
  for (int i = 1; i < argc; i++) {
    sum += atoi(argv[i]);
  }
  printf("%d\n", sum);
}
```

7. (10 points) Write a C function

   `double max(int n, double a[n]) { ... }`

   that returns the largest number in the array.

```
double max(int n, double a[n]) {
  double m = a[0];
  for (int i = 0; i < n; i++) {
    if (a[i] > m) {
      m = a[i];
    }
  }
  return m;
}
```

8. (10 points) Write a complete C program that takes a single command line argument, the name of a file containing integers, one per line, and prints the count of the even numbers from the file and the count of the odd numbers from the file. Do appropriate error checking.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>

int main(int argc, char *argv[]) {
  if (argc != 2) {
    fprintf(stderr, "usage: count file\n");
    exit(1);
  }

  FILE *f = fopen(argv[1], "r");
  if (f == NULL) {
    fprintf(stderr, "count: %s: %s\n", argv[1], strerror(errno));
    exit(1);
  }

  int n;
  int evencount = 0;
  int oddcount = 0;

  while (fscanf(f, "%d", &n) == 1) {
    if (n % 2 == 0) {
      evencount++;
    }
    else {
      oddcount++;
    }
  }
  printf("%d even\n", evencount);
  printf("%d odd\n", oddcount);
}
```

9. (10 points (bonus)) Write a function

   `void rotate(int n, int a[n]) { ... }`

   that rotates the items in the array one spot to the right. The rightmost item is moved all the way to the left. For example, if the array contains

1	3	5	7	9

   then after calling the function it will contain

9	1	3	5	7

   Your function has to work for arrays of any size.

```
int rotate(int n, int a[n]) {
  int tmp = a[n-1];
  for (int i = n - 2; i >= 0; i--) {
    a[i+1] = a[i];
  }
  a[0] = tmp;
}
```