Student Name:..... NID:.....

COP3223 Sec1: Fall'08 C Some Practice for Final Exam (170points)

NOTE THAT THESE QUESTIONS MUST BE COMBINED WITH ALL QUESTIONS FROM TEST 4 (AND ITS PRACTICE VERSION), QUESTIONS 1, 3 AND 4 FROM TEST 3 (AND QUESTION 2 FROM PRACTICE TEST 3), AND QUESTIONS 1 AND 4 FROM TEST 2 TO GET A REAL SAMPLE OF THE TYPES OF QUESTIONS ON THE ACTUAL FINAL EXAM.

1. (20 points) Assume input file **day.txt** contains the names of the days of the week: Sunday Monday Tuesday Wednesday Thursday Friday Saturday. Each string (a day name) is on a separate line.
   Assume input file hightemperature.txt contains `84 92 95 89 93 96 91` each on a separate line
   Assume input file lowtemperature.txt contains `55 54 53 59 61 52 61` each on a separate line

   What is the output of this program?

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

#define NUM_ITEMS 7
#define MAX_LENGTH 15

struct daydata{
        char dayname[MAX_LENGTH];
        int hightemp;
        int lowtemp;
        int temperaturerange;
};

typedef struct daydata daydata;

void get_data(*alldata);
FILE *gfopen(char[], char[]);
void list_all(*alldata);

int main()
{
        char tempstring[MAX_LENGTH];
        int input, i;
        daydata alldata[NUM_ITEMS];

        get_data(alldata);

        for(i = 0; i < NUM_ITEMS; i++)
          alldata[i].temperaturerange= alldata[i].hightemp - alldata[i].lowtemp;

        list_all(alldata);
}

void get_data(daydata *alldata)

{       FILE *fp;
        int i;
```

```
        fp = gfopen("day.txt", "r");
        for(i = 0; i < NUM_ITEMS; i++)
                fscanf(fp, "%s", alldata[i].dayname);
        fclose(fp);

        fp = gfopen("hightemperature.txt", "r");
        for(i = 0; i < NUM_ITEMS; i++)
                fscanf(fp, "%d", &alldata[i].hightemp);
        fclose(fp);

        fp = gfopen("lowtemperature.txt", "r");
        for(i = 0; i < NUM_ITEMS; i++)
                fscanf(fp, "%d", &alldata[i].lowtemp);
}

// graceful file open function
FILE * gfopen(char name[], char mode[])
{
        FILE *fp;
        fp = fopen(name, mode);
        if(fp == NULL)
        {
                printf("Error opening file %s, aborting\n", name);
                exit(1);
        }
        return fp;
}

void list_all(daydata *alldata)
{
        int i;
        printf("%-10s\tTemperatureRange\n","Day Name");
        for(i = 0; i < NUM_ITEMS; i++)
           printf("%-10s\t%5d\n", alldata[i].dayname, alldata[i].temperaturerange);
}
```

2. (15 points) Suppose you have this structure:

```
struct gas {
            float distance;
            float gals;
            float mpg;
       };
```

A. Write ONLY the function that takes a `struct gas` argument. Assume that the passed structure contains the `distance` and `gals` information. The function should calculate the correct value for the `mpg` field and return (through the function name) the now completed structure.

B. Write only the function that takes the address of a `struct gas` argument. Assume that the passed structure contains the `distance` and `gals` information. The function should calculate the correct value for the `mpg` field and assign it to the appropriate field, so that when the function has ended, the now completed struct will be available to the caller.

3. (20 points) Write a program that writes to output 1000 random 'words', separated by spaces (word simply means a sequence of alphabetical letters, not necessarily an actual word that can be found in the dictionary). The 'words' you write should be randomly generated sequences of lowercase letters, with random lengths between 3 and 9 characters.

4. (20 points) What is the output of this program?

```
#include <stdio.h>

struct foo{
        int num;
        char *word;
        struct foo *ptr;
};
void func1(struct foo);
void func2(struct foo*);
void func3(struct foo);

int main() {
        struct foo a;
        a.num = 5;
        a.word = "myword";
        func1(a);
        printf("1 %d %s\n", a.num, a.word);

        a.num = 100;
        a.word = "secondword";
        func2(&a);
        printf("2 %d %s\n", a.num, a.word);

        a.ptr = &a;
        a.num = 50;
        a.word = "mylastword";
        func3(a);
        printf("4 %d %s\n", a.num, a.word);
}

void func1(struct foo a)
{
        while(*(a.word) != '\0')
           {
                putchar(*(a.word));
                a.word++;
           }
        putchar('\n');
        if(a.num % 10 != 0)
           { a.num *= 2; }
        a.word--;
        printf("num is %d\n", a.num);
}

void func2(struct foo *a)
{
        while(*(a->word) != '\0')
           {
                putchar(*(a->word));
                a->word++;
```

```
           }
        putchar('\n');
        if(a->num % 10 != 0)
           { a->num *= 2; }
        a->word--;
        printf("num is %d\n", (*a).num);
}

void func3(struct foo a)
{
        if(a.num > a.ptr->num)
          { a.num = 500; }
        else
          { a.num = a.ptr->num + 1; }

        a.word = "myotherword";
        a.ptr->word = "yetanotherword";
        printf("3  %d %s\n", a.num, a.word);
}
```