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

COP3223 Sec2: Spr '11 C Some Practice for Final Exam (170points)

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

1. (12 points) A baseball player's batting average is calculated as the number of hits divided by the official number of at-bats. In calculating official at-bats, walks, sacrifices, and occasions when hit by the pitch are not counted. Write a program that takes input containing N, and then N player numbers and batting records. Each player's record ends with an asterisk. Trips to the plate are coded in the batting record as follows: H–hit, O–out, W–walk, S–sacrifice, P–hit by pitch. The program should output for each player the input data followed by the batting average.

   ```
   Sample input file:

   3
   12 HOOOWSHHOOHPWWHO*
   4 OSOHHHWWOHOHOOO*
   7 WPOHOOHWOHHOWOO*

   Output:

   Player 12's record: HOOOWSHHOOHPWWHO
   Player 12's batting average 0.4555

   Player 4's record: OSOHHHWWOHOHOOO
   Player 4's batting average 0.417

   Player 7's record: WPOHOOHWOHHOWOO
   Player 7's batting average 0.364
   ```

2. (15 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);
}
```

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. Your program must save the word as a string and then print it out using the percent-s format in printf.

4. (15 points) Define a structure type to represent a word list. The structure will contain one string component for the language of the words (e.g., English, Japanese, Spanish), and integer component that keeps track of how many words are in the list, and an arrayof `MAX_WORDS` 20-character strings to hold the words. Define the following functions to work with word lists:

   A. `add_word`: Takes as parameters a word and a wordlist structure to modify. If the wordlist is already full, it displays the message "List full, word not added"). If the word is already in the list, it leaves the structure unchanged. Otherwise, it adds the word to the list and updates the list size. Do not bother keeping the list in order.

   B. `contains`: Takes as parameters a word and a wordlist. If the word matches one of the wordlist entries, the function returns true, otherwise false.

   C. `equal_lists`: Takes two wordlists as parameters and returns true if the lists are in the same language, have the same number of elements, and every element of one list is found in the other (Hint: need to call `contains` a lot.

   D. `display_word_list` Displays all the words of its wordlist structure parameter (one word per output line).

5. (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);
}
```