EE 312 Spring 2017 Exam 1 -- Solutions

Problem Number	Question	Points Possible
1	Struct programming	34
2-4	Stack trace	12
5	Big-O	12
6	Short Answer	16
	SCORE OUT OF: 74	

1. You may break problems up into smaller methods. (In other words you may add helper methods wherever you like.)
2. You may not use functions that are not in the standard C library, except for the ones your write, of course. You may assume that stdlib.h, stdio.h and string.h are imported.

Here are some function signatures etc:

```
void* malloc (size_t size); // size in bytes
void free (void* ptr);
void* realloc (void* ptr, size_t size);
char* strcpy ( char * destination, const char * source );
printf -- %d (int), %u (unsigned int), %g (double/float), %s (string)
```

`int strcmp(const char *str1, const char *str2);` If Return value < 0 then it indicates `str1` is less than `str2`. If Return value > 0 then it indicates `str2` is less than `str1`. If Return value = 0 then it indicates `str1` is equal to str2.
Assume that `ints`, pointers, and one word in memory are 32 bits wide.

1. (8 pts)

```
Person** getPersonArray(char* names[], int n) {
    Person **people;
    int i = 0;
    int j = 0;

    people = (Person**) malloc(n / 2 * sizeof(Person*));

    for (i = 0; i < n; i += 2) {
        people[j] = (Person*) malloc (sizeof(Person));
        strcpy(people[j]->first, names[i]);
        strcpy(people[j]->last, names[i+1]);
        j++;
    }
    return people;
}
```

Grading Rubric for 8 points:

1 point – declare return people type correctly.
2 points – malloc the array (argument to malloc, cast return type correct)
1 point – loop to malloc individual person spaces, with correct values
1 point -- malloc space for individual persons
2 points – copy last and first names to each person using strcpy. If they changed the order, no penalty.
1 point – return people
-1 for up to 3-5 minor syntax errors, and so on.

Use your judgment and change the rubric for different kinds of answers. Give some free points if they know what they are doing.

a) (8 pts)

```
void freePersonArray(Person** people, int m) {
    int i = 0;

    for (i = 0; i < m; i++) {
        free(people[i]);
    }
    free(people);
}
```

Grading Rubric for 8 points:
2 points – loop for freeing people individually, with correct loop variables.
3 points – free individuals, indexed correctly
3 points – free people
-2 points for freeing in the wrong order.
-1 for up to 3-5 minor syntax errors, and so on.

b) (8 pts)*

```
int removeDuplicates(Person** people, int m) {
    int *duplicates = (int*) calloc(m, sizeof(int));
    int i, j;

    /* Find duplicates */
    for (i = 0; i < m - 1; i++) {
        for (j = i+1; j < m; j++) {
            if (strcmp(people[i]->last, people[j]->last) == 0) {
                duplicates[i] = 1;
                duplicates[j] = 1;
            }
        }
    }

    /** Free duplicates */
```

```
    for (i = 0; i < m; i++) {
        if (duplicates[i] == 1) {
            free(people[i]);
            people[i] = NULL;
        }
    }

    /* Put remaining elements next to each other */
    for (i = 0, j = 0; i < m; i++) {
        if (duplicates[i] == 0) {
            /* Here if it is not a duplicate */
            people[j++] = people[i];
        }
    }

    free(duplicates);

    return j;
}
```

Grading Rubric for 8 points:
2 points – for finding locations of duplicates.
2 points – for freeing duplicate names' locations.
2 points – for rearranging people to remove gaps.
2 points – for counting and returning value
-1 points for memory leaks. -1 for up to 3-5 minor syntax errors, and so on.

c) (10 pts)

```
Person* findFirstInAPhoneBook(Person **people, int m) {
    int i = 0;
    Person *first = people[0];

    for (i = 1; i < m; i++) {
        if (strcmp(first->last, people[i]->last) > 0) {
            first = people[i];
        }
    }
    return first;
}
```

Grading Rubric for 10 points:
2 points – for declaring and initializing variable to hold first person. Partial OK if they got pointer or dereferencing wrong.
1 points – loop through people, with correct index.
5 points – for comparing each last name to person.
2 points – for returning pointer to first person.
-1 for up to 3-5 minor syntax errors, and so on.

2. (4 pts) partial OK
```
17, 7
```

3. (4 pts) partial OK
```
Man Sam Man
```

(4 pts)* partial OK.
```
01201
```

5. (12 pts total, 2 pts each)
Grading Rubric: 1 pt for answer, 1 for explanation
a.
O(1)
This function has a fixed size input and will not take more time of different inputs.

b.
O(n^2)
The inner while loop loops i times, and the outer one loops n times.
i goes from 1 to n, so the function's runtime would be proportional to 0.5n^2. This function is O(n^2).

c.
O(N)
1 + 2 + 4 + 8 + ... + N
(1 - N) / (1 - 2) = N - 1 = O(N)

d.
O(n)
Worst case, no elements are equal to i, so it adds 3 to i every iteration. This would take O(n) time.

e.
But the runtime is O(n^2). Outer loop goes n times, inner loop goes j times, overall n^2 runtime.

f.
O(log^2(N))
The outer loop will run in log(sqrt(N)) time. This is O(log(N)) time. The inner modification to k will make it take that loop squared, so the overall runtime is log squared.

6. 4 points each
```
a. (C)
b. free(p); // at the very end of the program
c. ii, v
d. int *p = malloc(sizeof(int)); free(p); free(p);
```