EE 312 Spring 2017 Exam 1

I promise that all work on this test is my own, that I have received no outside assistance on it, and that I am adhering to the university honor code.

Your Name_____ Your UTEID_____ Date_____

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

Instructions:

1. Please turn off your cell phones.
2. You have 120 minutes to complete the test.
3. You may not use a calculator.
4. There is scratch paper at the end. You may ask for additional scratch paper if required. If your actual answer is on the scratch paper, state that fact clearly and legibly in the space below the question.
5. When code is required, write C code.
6. Problems marked with * may require more time, so plan accordingly.
7. You may break problems up into smaller methods. (In other words you may add helper methods wherever you like.)
8. 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.
9. Style is not evaluated when grading, but neatness and legibility are required. Use indentation to help the graders read your code.
10. The proctors will not answer most questions. Write down any assumptions that you need to make to resolve your doubts.
11. When you finish, show us your UT ID, turn in the exam and all scratch paper. Use the stapler we have if required.

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.

To find the sum of a finite geometric series, use this formula:

$Sn = a_1(1 - r^n) / (1 - r), r!=1$

where n is the number of terms, a_1 is the first term and r is the constant factor between consecutive terms.

To find the sum of a finite arithmetic series, use this formula:

$Sn = n (t_1 + t_n) / 2$

Where n is the number of terms, t_1 is the first terms and t_2 is the last term.

---

1. (8 pts) Assume that the following struct is defined

```
typedef struct person {
    char first [11];
    char last [11];
} Person;
```

   a) Write a function `getPersonArray` that takes as input (i) an array of `char` pointers (`char* names[]`) representing null-terminated strings and (ii) the number of strings in the array, `int n`, and organizes them into an array of `struct` pointers on the heap. `getPersonArray` returns a pointer to an array of pointers, each element of the array pointing to a `Person` struct on the heap.

   The struct array should therefore be organized as an array of pointers to structs. Hint: when allocating memory on the heap, recall the two-step process used in the Matrix project/assignment.

`names` is organized in the following manner:

- `names` is never null, or empty.
- `n` >= 2, and `n` is even.
- Each string in `names` has only uppercase and lowercase letters, and itself will never be empty or null.
- Each string's length <= 10.
- The strings in `names` are always in pairs – the first in each pair represents the last name, and the second represents the first name.

Example: If `myNames is ["Smith", "Jake", "Jones", "Mary"],` the pointers in the array returned by `getPersonArray(myNames, 2)` would contain an array with two elements; the first element would be { “Jake”, “Smith” } and the second element would be { “Mary”, “Jones”}.

```
Person** getPersonArray(char* names[], int n) {
```

b) (8 pts) Write a function `freePersonArray` that takes as input the `Person**` array returned by the function in part a) and frees all the memory. `m` is the number of `Persons` in `people`.

```
void freePersonArray(Person** people, int m) {
```

c) (8 pts)* Write a function that deletes each person that does not have a unique family name. (Everybody without a unique last name in the original array should be removed.) The function returns the number of people that remain in the array. Make sure to free the memory for people that are not in the array any longer. (Make sure to change the values of pointers to avoid double-delete if `freePersonArray` is invoked later.) People that remain in the array should be next to each other starting from index 0. `m` is the number of `Persons` in `people`.

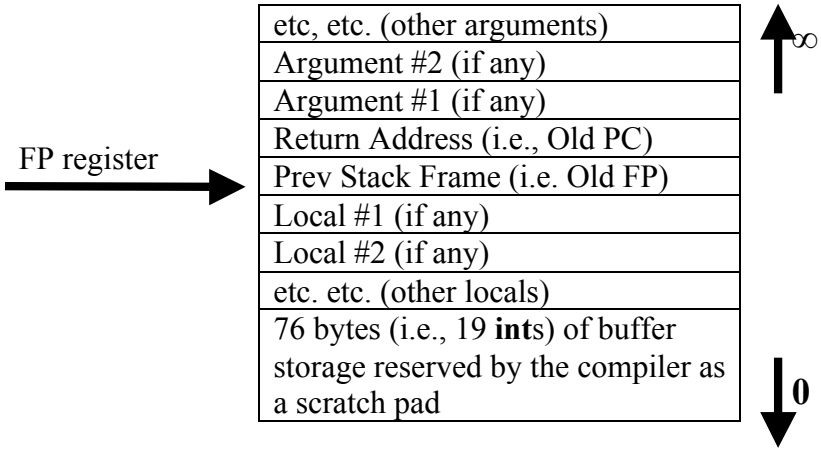
```
int removeDuplicates(Person** people, int m) {
```

d) (10 pts) Write a function that returns a pointer to a person that would be the first in a phone book based on the last name (consider lexicographical order). If multiple people have the same last name (which would be the first in the lexicographical order), return the first such person from the given array. `m` is the number of `Persons` in `people`.

```
Person* findFirstInAPhoneBook(Person **people, int m) {
```

Section 2: Program Analysis

For the questions in this section, analyze the program and indicate the output produced when the program is run. There is no partial credit in this section, so please be very careful. NOTE: all of the programs do run (there are no syntax errors or other compiler errors). You should assume that the following stencil is used to organize the information contained in a stack frame.



Stencil used by Visual Studio for a Stack Frame in C programs. (Stack Frame was discussed extensively in lecture and recitation). Stack Frames are placed at successively smaller addresses as functions are called (i.e., the stack grows down).

2. (4 pts) What is the output of the following program?

```
void doit(int32_t a, int32_t b[]) {
  a = 41;
  b[0] = 7;
  b[1] = 17;
}

int main(void) {
  int32_t a = 42;
  int32_t b[1] = {10};
  doit(a, b);
  printf("%d %d", a, b[0]);
}
```

_____

3. (4 pts) What is the output of the following program?

```
void doit(char x[], char y[]) {
  x = y;
  x[3] = ' '; // space character, ASCII value 32
}

int main(void) {
  char x[4] = "Man";
  char y[4] = "Sam";
  doit(x, y);
  printf("%s %s", x, y);
}
```

_____

4. (4 pts)* What is the output of the following program.

```
void main() {
    int i, *a[5], **b;
    for (i=0; i<5; i++) {
        a[i]=(int*) malloc(sizeof(int));
        *a[i]=i%2;
    }
    i--;
    for (b=a+i; i>0; i--) {**(b-i)=**b+i%3;}
    for (i=0; i<5; i++) { printf("%d",*b[-i]); free(b[-i]); }
}
```

5. (12 pts total, 2 pts each) Give an expression in "big-Oh" notation for the amount of time the following doit functions will take in the *worst case*. Also add 1-2 sentences that explains the reasoning behind your answer. The space for answers is ABOVE each code snippet.

a. Ans:_____

Explanation: _____

_____

```
void doit(char n){
    printf("n is %c\n", n);
}
```

b. Ans:_____

Explanation: _____

_____

```
int doit(int n)
    int i = n;
    int j;
    while(i > 0) {
        j = i;
        while(j > 0) {
            j = j - 1;
        }
        i = i - 1;
     }
     return n*n*n;
}
```

c. Ans:_____

Explanation: _____

_____

```
int doit(int N) {
   for (int j = 1; j < N; j *= 2) {
      for (int k = 1; k < j; k += 1) {
      }
   }
}
```

d. Ans: _____

Explanation: _____

_____

```
int doit(int* array, int n){
        int i = 1;
       int x = 0;
        while(i < n){
                if(array[i] == 0){
                        x = x + 1;
                        i = i * 2;
                } else {
                        i = i + 3;
                }
        }
        return x;
}
```

e. Ans: _____

Explanation: _____

_____

```
void doit(int n){
   while(n > 0){
            int j = n;
           while(j > 0){
                   funMe(); // funMe is O(1)
                   j = j - 1;
           }
           n = n - 1;
   }
}
```

f. *Ans_____

Explanation: _____

_____

```
int doit(int N) {
   int start = 1;
   int k;
   for (k = 1; k < N; k *= 2) {
      if (k * k >= N) {
         start = start * 2;
         k = start;
      }
    }
}
```

Section 3: Semi-Short Answers

6. 4 points each
   a. Choose a statement to replace "#####" (from the given options) such that the given program prints a string (read from the standard input) in the following way: last character, first character, second to the last character, second character, etc. Each character should be printed only once.

```
void main() {
  char string[10], *p, *k;
  scanf("%s", string);
  p = k = string;
  while (*(k+1)) k++;
  #####
```

(A)
```
while (p < k) {
  printf("%c", *k--);
  if (k != p)
    printf("%c", *p++);}
```

(B)
```
for (; p < k; p++, k--){
  printf("%c", *k);
  printf("%c", *p);}
```

(C)
```
while ( p<=k ) {
  if (k != p)
    printf("%c", *k--);
  printf("%c", *p++);}
```

   b. The following program has at least one memory leak. Please fix it by adding line(s) of code.

```
int main(void) {

    int* p = (int*)malloc(14 * sizeof(char));

    char* q = (char*)p;

    strcpy(q, "random string");

    *p = 65;

}
```

c. Choose all the statements that are true for the C programming language by circling the number on the left iff the corresponding statement is true.

   i. If `c` and `t` variables are declared as `char *c, t[20];` the following assignment is legal: `c = t;`
   ii. If a is declared as an array of ints: `int a[10]`, then `sizeof(a)` returns the number of elements in the array.
   iii. It is legal to declare a function inside another function.
   iv. If `e` is declared as an array of `ints: int e[10]`, `e[9]` is the same as `*(e + 9)`.

d. Using a single variable and one invocation to `malloc`, illustrate double-delete error.

Scratch paper

Scratch paper