ENEE 140, Spring 2017
Final Exam — Answer Key
Do Not Make a Copy!!

Date:

University of Maryland Honor Pledge: The University is committed to Academic Integrity, and has a nationally recognized Honor Code, administered by the Student Honor Council. In an effort to affirm a community of trust, the Student Honor Council proposed and the University Senate approved an Honor Pledge. The University of Maryland Honor Pledge Reads:

"*I pledge on my honor that I have not given or received any unauthorized assistance on this examination (or assignment)*"

Please write the exact wording of the Pledge, followed by your signature, in the space below:

Pledge: _____
Pledge: _____
Pledge: _____
Pledge: _____

Your signature: _____
Full name: _____ Course: _____ Directory ID: _____

List of Exam Questions:

Question:	1	2	3	4	5	6	7	8	Total
Points:	16	16	10	15	9	6	12	16	100
Score:									

Instructions:

- Make sure that your exam is not missing any sheets, then write your full name, your section and your Directory ID on the front.
- Write your answers in the space provided below the problem. If you make a mess, clearly indicate your final answer.

- The exam has a maximum score of 100 points.
- The problems are of varying difficulty. The point value of each problem is indicated. Pile up the easy points quickly and then come back to the harder problems.
- This exam is OPEN BOOK. You may use any books or notes you like. Calculators are allowed, but no other electronic devices. Good luck!

1. (16 points) This problem tests your understanding of C types and casts and of C operators. Assume that variables `a`, `b`, `c` and `d` are defined as follows:

```
int      a = -1;
unsigned b = 1;
float    c = 2;
float    d = 3;
```

Fill in all the empty cells in the table below. For each of the C assignment expressions in the left column, state the resulting value of the `r2`–`r9` variables. If an expression results in a compilation error, write ERROR.

Assignment		Value
float	r1 = a / c;	-0.5
int	r2 = a / (**int**)d;	**0**
int	r3 = (a − b) < 0;	**0**
int	r4 = INT_MAX + INT_MIN;	**-1**
unsigned	r5 = (**unsigned**)a % 2;	**1**
unsigned	r6 = c % d;	**ERROR**
int	r7 = d / c;	**1**
float	r8 = b / 2 + d / 2;	**1.5**
int	r9 = b << 1 && 1;	**1**

2. (16 points) This question tests our understanding of files and character input/output. The following function will read from a file, swap the case of every letter and print out the results to a new file. Find and correct each of the bugs.

```
int swap_case(char in, char out)
{
    int c;
    FILE original = fopen(in, "r");

    if(original = NULL){
        printf("input file does not exist\n");
        return 1;
    }
    FILE swapped = fopen(out, "w");

    while( getc(original, "%c", &c)  != EOF )
    {
        if('a' >= c <= 'z'){
```

```
            c += 'A'-'a';
        }
        else if('A' >= c <= 'Z'){
            c -= 'A'-'a';
        }
        fputc(swapped, "%c", c);
    }

    return 0;
}
```

Solution:

```
int swap_case(char in[], char out[])      // need in[] or char*
{
    int c;
    FILE *original = fopen(in, "r");      // need FILE*

    if(original == NULL){                 // ==
        printf("input file does not exist\n");
        return 1;
    }
    FILE *swapped = fopen(out, "w");      // need FILE*

    while( (c = getc(original)) != EOF ) // used scanf format instead of getc
    {
        if(c >= 'a' &&  c <= 'z'){        // needs &&
            c += 'A'-'a';
        }
        else if(c >= 'A' &&  c <= 'Z'){  //needs &&
            c -= 'A'-'a';
        }
        fputc(c, swapped);                // used fscanf format instead of fputc
    }

    return 0;
}
```

3. (10 points) This problem tests your understanding of composite types and hashing. Write the output of the following program.

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

#define MAX_STRING 80
#define NUM_MOVIES 5
```

```
typedef struct movie {
    int year;
    char title[MAX_STRING];
} Movie;

Movie leos_movies[NUM_MOVIES];

int hash_function(int d) {
    int hashval;

    /* Stores the sum of the digits of d in hashval */
    /* For example, if d is 823, then hashval will be 8+2+3=13 */
    for (hashval = 0; d != 0; d = d / 10){
        hashval += (d % 10);
    }

    return hashval % NUM_MOVIES;
}

void insert(int key, char value[]) {
    int start_index = hash_function(key);
    int index = start_index;
    do {
        if (leos_movies[index].year == key) {
            /* Update value for this key */
            strcpy(leos_movies[index].title, value);
            return;
        } else if (leos_movies[index].year == 0) {
            /* Insert new key-value pair */
            leos_movies[index].year = key;
            strcpy(leos_movies[index].title, value);
            return;
        }
        index = (index + 1) % NUM_MOVIES;
    } while (index != start_index);
}

int main() {
    int i;

    for (i = 0; i < NUM_MOVIES; i++) {
        leos_movies[i].year = 0;
        strcpy(leos_movies[i].title, "");
    }

    insert(1997, "Titanic");
```

```
    insert(2006, "The Departed");
    insert(2010, "Shutter Island");
    insert(2010, "Inception");

    for (i = 0; i < NUM_MOVIES; i++) {
        if (leos_movies[i].year == 0) {
            printf("EMPTY\n");
        } else {
            printf("%s (%d)\n", leos_movies[i].title, leos_movies[i].year);
        }
    }

    return 0;
}
```

Solution:

```
EMPTY
Titanic (1997)
EMPTY
The Departed (2006)
Inception (2010)
```

4. (15 points) This problem tests your understanding of arrays and aggregate values. Fill in the code so that it finds the largest value in each column of a 4x4 matrix of unsigned integers and assigns them to a new 1x4 array. For example, given the following matrix:

```
8 5 6 9
2 7 4 1
7 3 5 4
4 1 6 2
```

The resulting row array should be:

```
8 7 6 9
```

```
void
maximum_columns(unsigned m[4][4]) {
    unsigned result[4];       /* Array to store column maximums in*/
    unsigned col_max = ____0____;
    int row, col;

    for (____col____ = 0; ____col____ < 4; ____col____++) {
        for (____row____ = 0; ____row____ < 4; ____row____++) {
            if (__m[row][col]__ > ____col_max____) {
                col_max = m[____row____][____col____];
```

```
            }
        }
        result[      col      ] =   col_max   ;
        col_max     =     0      ;
    }
}
```

Solution:

```
void maximum_columns(unsigned m[4][4]) {
    unsigned result[4];       /* Array to store column maximums in*/
    unsigned col_max = 0;
    int row, col;

    for (col = 0; col < 4; col++) {
        for (row = 0; row < 4; row++) {
            if (m[row][col] > col_max) {
                col_max = m[row][col];
            }
        }
        result[col] = col_max;
        col_max = 0;
    }
}
```

5. (9 points) This questions tests your understanding of string functions and manipulations. What is the output of the following program? (HINT: Draw out the strings as arrays and keep track of the changes to it as you step through the program)

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

int main() {
    int i;
    char string1[20] = "cat";
    char string2[20] = "dog";
    char string3[20] = "hotels";
    char string4[] = "bzz";

    strncat(string1, string2, 20);

    printf("%s\n", string1);

    for(i = 3; i < 6; i++) {
        string3[i] = string2[i-3];
```

```
    }

    printf("%s\n", string3);

    for(i = 0; i < strlen(string4); i++) {
        string4[i] = string4[i] - 1;
    }

    printf("%s", string4);

    return 0;
}
```

Solution:

```
catdog
hotdog
ayy
```

6. (6 points) Write an expression using `rand()` to generate a random number in each of the following ranges.

a) [−20, 48]
b) Even numbers up to 100
b) Multiples of 3 in the range [9, 30]

Solution:

a) `rand()%(48-(-20)+1) - 20`
b) `2*(rand()%(50 + 1))`
c) `3*(rand()%(10 - 3 + 1) + 3)`

7. (12 points) Write a program that uses a bit pattern stored in integer array `int a[32]` to create and print an unsigned integer `x` that has the same bit pattern as the array

for example if the array `a[32]` is set to `{0,0,0,.....,0,1,0,1}`, then the unsigned integer `x` should have the value 5.

```
#include <stdio.h>

int main(){
    int i;
    unsigned x;
    int a[32] = {0};
    a[31] = 1;
```

```
    a[29] = 1;

    // a is set to {0,0,0,.....,0,1,0,1}

    ____x=0____;
    for(i = 0; ____i<32____; ____i++____){
        if(a[i] == ____1____) {
            x = ____x | 1<<(31-i)____;
        }
    }


    printf("%u", x);
    return 0;

}
```

Solution:

```
#include <stdio.h>

int main(){
    int i;
    unsigned x;
    int a[32] = {0};
    a[31] = 1;
    a[29] = 1;

    // a is set to {0,0,0,.....,0,1,0,1}


    x = 0;
    for(i = 0; i < 32; i++){
        if(a[i] == 1) {
            x = x | 1<<(31-i);
        }
    }

    printf("%u", x);
    return 0;

}
```

8. (16 points) Write a C program to find and print the most frequently occurring character in a string. Assume that the number of unique characters that can be used is 255 (each with a unique ASCII value), and that the size of the string is 100 characters. Also, assume that there's only one character that occurs the most (there are no ties).

   For example, if the string is `"ENEE"`, the output should be: "the most frequently occurring character is E".

```
#include <stdio.h>
#define MAX_CHARS 255   //number of unique characters

int main()
{
    char string[100];
    // use this array to keep track of the frequency
    // (number of occurrences) of each character
    int frequency[MAX_CHARS] = {0};
    int i, max, ascii;

    printf("Enter any string: ");
    scanf("%s", string);


    /* Find the frequency of each character */
    i = ____0____;
    while(string[i] != ____'0'____) {
        ascii = ____(int)string[i]____;          // get the ascii value of the character
        frequency[ascii] ____++____;      // count how often the character occurs
        i++;
    }


    /* find the most occurring character  */
    max = 0;
    for(i=0; ____i<MAX_CHARS____; i++) {
        if(frequency[i] > ____frequency[max]____)
            ____max = i____;
    }


    printf("the most occurring character is %c", ____frequency[max]____);

    return 0;
}
```

Solution:

```
#include <stdio.h>
#define MAX_CHARS 255 // Maximum unique characters allowed

int main()
{
    char string[100];
    int frequency[MAX_CHARS] = {0};
    int i, max, ascii;

    printf("Enter any string: ");
    scanf("%s", string);


    //Find frequency of each characters
    i=0;
    while(string[i] != '\0') {
        ascii = (int)string[i];           // get the ascii value of the character
        frequency[ascii]++;               // count how many times the character occu
        i++;
    }


    // Finds maximum frequency
    max = 0;
    for(i=0; i<MAX_CHARS; i++) {
        if(frequency[i] > frequency[max])
        max = i;
    }


    printf("the most frequently occurring character is %c", frequency[max]);

    return 0;
}
```