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	Total
Points:	16	8	19	26	9	12	10	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 = -2;
float    b = -1;
unsigned c = 1;
float    d = 2;
```

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 = b / d;	-0.5
float	r2 = b / a;	**0.5**
unsigned	r3 = c % −a;	**1**
unsigned	r4 = d % −a;	**ERROR**
unsigned	r5 = (**int**)UINT_MAX + c;	**0**
unsigned	r6 = (**unsigned**)INT_MIN % 2;	**0**
float	r7 = c / a * (**int**)(d);	**0**
float	r8 = ++c / −a * (**int**)(d);	**2.0**
float	r9 = (**float**)(3 % −a) / a;	**-0.5**

2. (8 points) This problem tests your understanding of basic variable manipulations in C. The following function is supposed to prompt the user for an integer number and to print whether the number provided is negative, positive or 0. Find all the bugs in this implementation:

```
void
test_integer()
{
  printf("Enter any integer:");
  scanf("%d", num);

  if (num = 0) {
    printf("%d was zero!", &num);
  } else if (num < 0) {
    printf("You entered a negative number!");
  } else {
    printf("You entered a positive number!");
  }
}
```

Solution:

- Variable num not declared
- Variable rather than address provided to `scanf()`

- Assignment instead of equality test in if condition
- Address rather than variable provided to `printf()`

```
void
test_integer()
{
  int num;
  printf("Enter any integer:");
  scanf("%d", &num);

  if (num == 0) {
    printf("%d was zero!", num);
  } else if (num<0) {
    printf("You entered a negative number!");
  } else {
    printf("You entered a positive number!");
  }
}
```

3. (19 points) This two-part question tests your understanding of strings and loops.

   (a) Fill in the blanks for this program that determines if a string is a palindrome. A palindrome is a word that is read the same forwards and backwards, e.g. "kayak".

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

int main() {
  int i, length;
  char word[100]; // The word can have a maximum length of 100

  printf("Word: ");
  scanf("%s", word); // Reads the word into a string.

  length = strlen(word); // strlen() gets the length of a string.

  for (i = 0 ; i < length ; i++ ) {
    if (word[ i ] != word[ length-1-i ]) {
      printf("This word is not a palindrome.\n");
      return 0;
    }
  }

  printf("This word is a palindrome.\n");
  return 0;
```

}

Solution:

```
...
for (i = 0; i < length; i++) {
  // You only have to loop over half of the string,
  // so the exit condition may also be i < length / 2

    if (word[i] != word[length - 1 - i]) {
        ...
...
```

(b) Rewrite the `for` loop as a `while` loop.

Solution:

```
...
i = 0;
while (i < length) {
  // You have the same flexibility about the
  // exit condition as in the for loop.

    if (word[i] != word[length - 1 - i]) {
       printf("This word is not a palindrome.\n");
       return 0;
    }
    i++;
}
...
```

4. (26 points) This question tests your understanding of random number generation, formatted I/O and conditional statements. The following program generates a random floating-point cost in the range [0, 100]. The user is assumed to only be in possession of $10 bills and is prompted for the number of bills they would like to pay with. If the correct amount or greater was provided, the amount of change is calculated and printed. Otherwise, the amount of money that is still required is printed instead. The dollar amounts should be printed to within 2 decimal places.

   Fill in the blanks to complete the program. You can use the function `rand()` to generate random integers and assume that the largest possible random integer is given by `RAND_MAX`. The random number generator is seeded for you.

```
int main() {

    srand(time(NULL)); // seeds random number generator
    float cost;
    int tens;

    // compute random floating-point cost from [0, 100]
    cost = ((float) ____rand()____ / ____RAND_MAX____ ) * (____100____);

    // print out the cost
    printf("The total is: $____%.2f____ \n", ____cost____);
    printf("How many $10 bills are you paying with? ");

    // prompt user for amount of $10 bills and store in "tens"
    scanf("____%d____", ____&tens____);

    // print out amount of change if enough money was provided
    // otherwise, print out the amount the user is short on
    if(____tens*10____ >= ____cost____)
        printf("Your change is: $____%.2f____\n", ____tens*10 - cost____);
    else
        printf("$____%.2f____ is still required \n", ____cost - tens*10____);
}
```

Solution:

```
int main() {

    srand(time(NULL)); // seeds random number generator
    float cost;
    int tens;

    // compute random floating-point cost from [0, 100]
```

```
    cost = ((float)rand()/RAND_MAX) * (100);

    // print out the cost
    printf("The total is: $%.2f \n", cost);
    printf("How many $10 bills are you paying with? ");

    // prompt user for amount of $10 bills and store it in "tens"
    scanf("%d", &tens);

    // print out amount of change if enough money was provided
    // otherwise, print out the amount the user is short on
    if(tens*10 >= cost)
        printf("Your change is: $%.2f\n", tens*10 - cost);
    else
        printf("$%.2f is still required\n", cost - tens*10);
}
```

5. (9 points) This problem tests your understanding of functions and local variables. What is the output of the following program?

```
int main() {
    int a = 100;
    printf("func(a)*a = %d \n", func(a)*a);
    printf("a = %d \n", a);
}

int func(int p) {
    int a = 20;
    printf("a*p = %d \n", a*p);
    a = a*5;
    return a;
}
```

Solution:

```
a*p = 2000
func(a)*a = 10000
a = 100
```

6. (12 points) This problem tests your understanding of arrays. Write a function that returns the average of a sub-sequence of an array. The function takes as parameters an integer array of 10 elements and two integers `L` and `U`. The integers `L` and `U` are the indices of the lower and upper bounds of the sub-sequence. The function should return the average of all the integers between and including the elements on positions `L` and `U` in the array. You may assume that `L` < `U`. For example, given the array:

```
2 2 7 8 1 6 7 7 8 9
L = 2 and U = 5
```

The average is calculated as follows `(7 + 8 + 1 + 6)/ 4`

```
float
func1( int array1[ ] , int L, int U){
    int sum,count,i;
    float avg;

    sum = 0;
    count = 0;
    for( i=L ; i < U+1 ; i++){
        sum = sum + array1[i];
        count++;
    }

    avg = (float)sum/count;
    return avg;
}
```

Solution:

```
#include <stdio.h>
  float func1(int array1[], int L, int U){
    int sum,i,count;
    float avg;

    sum = 0;
    count = 0;
    for(i = L; i < U+1; i++){
      sum = sum + array1[i];
      count++;
    }

    avg = (float)sum/count;

    return avg;
  }
```

```
int main(){
  int a[10] = {2, 2, 4, 5, 2 , 5, 5, 8, 9 ,1};
  int L = 2, U = 5;
  printf("%.2f", func1(a,L,U));

  return 0;
}
```

7. (10 points) This problem tests your understanding of the binary representation of numbers. Write a function that takes an unsigned integer and prints the number of 1's in its binary representation. For example, if `num = 5` the function should return 2.

```
int count1s(unsigned int num)
{
    unsigned totalBits = ____sizeof(num)____ *8;
    int count, i;

    for (i = 0; ____i < totalBits____ ; ____i++____ ) {
        if (____(num & (1 << i)) == 1____)
            count++;
    }

    return ____count____;
}
```

Solution:

```
int count1s(unsigned int num)
{
    unsigned totalBits = sizeof(num)*8;
    int count = 0;
    int i;

    for(i=0;i< totalBits;i++)
    {
        if( num & (1<< i) )
            count++;
    }

    return count;
```

```
}
```