ENEE 140, Spring 2019
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	10	15	9	12	16	14	8	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:

```
float       a=3;
char        b='a';
unsigned    c=4;
int         d=6;
```

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**
int	r0= ('1'−'0');	1
unsigned	r1=UINT_MAX+12;	**11**
float	r2=a%4;	**ERROR**
int	r3= INT_MIN==(−INT_MIN);	**1**
char	r4=b−('a'−'A');	**'A'**
int	r5=(INT_MAX+1)−INT_MIN;	**0**
unsigned	r6=++d − d++;	**0**
char	r7=b+a;	**'d'**
float	r8=c++%3;	**1**

2. (10 points) This problem tests your understanding of command line arguments. The program will be executed with the following command line:

   ```
   a.out num1 num2 num3 num4 num5
   ```

   `num1`, `num2`, `num3`, `num4`, and `num5` are integers that the user will enter on the command line. Fill in the blanks of the following program so that all the values from the command line are put into an array of integers. The program should then take this array and print out the largest value.

   An example execution of this program would be:

   ```
   a.out 34 2341 546 23 1
   The largest value is 2341!
   ```

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


   int main(int argc, char *argv[]){


   int numArray[_____5_____];
   int i;
   int max=__INT_MIN__;
           for(i=0;i<5;i++){
                   numArray[i]=atoi(__argv[i+1]__);
           }
           for(i=0;i<5;i++){
                   if(__numArray[i] >__ max){
                           max=numArray[i];
                   }
           }
           printf("The largest value is %d\n", ____max____);
           return 0;
   }
   ```

   Solution:

   ```
   SOLUTION:
   #include <stdio.h>
   #include <stdlib.h>


   int main(int argc, char *argv[]){
   ```

```
int numArray[5]; //This can be any value less than or equal to 5.
int i;
int max=INT_MIN;
        for(i=0;i<5;i++){
                numArray[i]=atoi(argv[i+1]);
        }
        for(i=0;i<5;i++){
                if(numArray[i]>max){
                        max=numArray[i];
                }
        }
        printf("The largest value is %d\n",max);
        return 0;
}
```

3. (15 points) This problem tests your understanding of random number generation, command line arguments, and file I/O.

   The following program is supposed to open the file specified by the first command line argument, then prompt the user for two integers. The program should then generate a sequence of 10 random integers between AND including those two numbers and write each number to the file.

   Correct all the errors in the code, both syntax errors AND errors that will prevent the program from doing exactly what is specified in the paragraph above.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(int argc, char * argv[]) {

    srand(0);

    FILE file_name;
    int i, val, max, min;

    file_name = fopen(argc[1], "w");

    if(file_name){
        printf("Couldn't open file with name %s", file_name);
        exit(-1);
    }

    printf("\nEnter the maximum integer to generate: ");
    scanf("%d", &max);
    printf("\nEnter the minimum integer to generate: ");
    scanf("%d", &min);

    for(i = 0; i <=10; i++){
        val = min + (rand() % (max - min));
        printf(file_name, "\n%d", val);
    }

    fclose(argv[1]);
    return 0;
}
```

Solution:

```
//Solution
  int main(int argc, char * argv[]) {

    srand(time(0)); //Error 1: instead of srand(0)

    FILE * file_name; //Error 2: instead of FILE
    int i, val, max, min;

    file_name = fopen(argv[1], "w");     //Error 3: argv[1], not argc[1]

    if(!file_name){ //Error 4: !file_name
        printf("Couldn't open file with name %s", argv[1]); //Error 5: Print argv
```

```
                                                                //instead of file pointer
        exit(-1);
    }

    printf("\nEnter the maximum number to generate: ");
    scanf("%d", &max);
    printf("\nEnter the minimum number to generate: ");
    scanf("%d", &min);

    for(i = 0; i <10; i++){    //Error 6: Runs 11 times, not 10 times
        val = min + (rand() % (max - min + 1)); //Error 7: generates the correct
                                                //range of numbers
        fprintf(file_name, "\n%d", val); //Error 8: fprintf, not printf
    }

    fclose(file_name);  //Error 9: Close the file pointer, not the name
    return 0;
}
```

4. (9 points) This problem tests your understanding of random number generation. Given the following specifications, write in a single line a statement that will generate random numbers according to each specification. You may assume that the random number generated was properly seeded and you may use UINT_MAX as the maximum **unsigned** value.

   (a) An integer from 0 to 140

   Solution: rand() % 141

   (b) A real number between -9.5 and 10.5

   Solution: −9.5 + (**float**)rand()/UINT_MAX ∗ (20)

   (c) An odd number between 26 and 60

   Solution: 27 + 2∗(rand()%17)

5. (12 points) This problem tests your understanding of command line arguments and strings. What is the output of this program when it is compiled as `char_erase.out` and executed on the command line as follows?

```
char_erase.out Have a great summer vacation!
```

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

```
int main(int argc, char* argv[]) {
    int x, L;
    printf("argc␣=␣%d\n", argc);
    for (x=0; x < argc; x++) {
        L = strlen(argv[x]);
        argv[x][3*x % L] = '_';
        printf("argv[%d]␣=␣\"%s\"\n", x, argv[x]);
    }
    return 0;
}
```

Solution:

```
argc = 6
argv[0] = "_har_erase.out"
argv[1] = "Have_"
argv[2] = "_"
argv[3] = "grea_"
argv[4] = "_ummer"
argv[5] = "vacati_n!"
```

6. (16 points) This problem tests your understanding of functions and characters.

Complete the program such that it will read in a line of text from the user (which may have letters, numbers, spaces and punctuation), and the prints out the "Flip Cipher" encoded line.

To apply the "Flip Cipher," you need to switch all letters with the "opposite letter" of the alphabet. For example, `'A'` becomes `'Z'`, and `'y'` becomes `'b'`. For example, here is an example of an output for a given input.

Input: `This is Exam 2!`
Output: `Gsrh rh Vczn 2!`

Fill in the blanks to implement this functionality.

```
____char____ flip_cipher(char c){
    if(c <= 'z' ____&&____ c >='a') {
        return 'a' + ('z' - c);
    }
    else if(c <= 'Z' ____&&____ c >='A'){
        return 'A' + ('Z' - c);
    }else{
        return ____c____;
    }
}
```

```
int main(){
    char in;

    while((in = __getchar()__)!='\n'){
        printf("%c", flip_cipher(____in____));
    }
    return 0;
}
```

Solution:

```
Problem 2 Solution:

//Solution
#include<stdio.h>

char flip_cipher(char c){
    if(c <= 'z' && c >= 'a') {
        return 'a' + ('z' - c);  //'z' - ('a' - c) is also acceptable
    }
    else if(c <= 'Z' && c >= 'A'){
        return 'A' + ('Z' - c); //'Z' - ('A' - c) is also acceptable
    }else{
        return c;
    }
}

int main(){
    char in;

    while((in = getchar())!='\n'){
        printf("%c", flip_cipher(in));
    }
```

7. (14 points) This problem tests your understanding of file manipulation and functions. The following program is intended to read a nine-digit number from a file character by character and then directly convert it into its integer representation through a process of summing the digits and multiplying by an appropriate power of 10. You may assume that the file contains has exactly 9 digits and no other characters.

   This value will then be printed out to complete the program. The program will be executed with the line:

   `a.out file1.txt`

   Please find all the bugs in the implementation below and indicate the correction necessary for each error.

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

void power(int number, int exp);

int main(int argc,char *argv[]){

        FILE *fp=fopen(argv[0],"r");
        int x;
        int i=8;
        int sum=0;
        char c;
        while((c=fgetc(fp))!=EOF){
                x=c;
                sum=sum+x*power(x,i);
                i--;
        }
        printf("The value is %d", sum);
}

int power(int number, int exp){
        int i;
        int value=0;
        for(i=0;i<exp;i++){

                value=value*number;
        }
        return number;
}
```

Solution:

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

void power(int number, int exp);//needs a return type
int main(int argc,char *argv[]){

        // Must test whether the argument was provided, e.g.
        //              if (argc<2) return -1;
        FILE *fp=fopen(argv[0],"r");//Should be argv[1]
        int x;
        int i=8;
        int sum=0;
        char c;
        while((c=fgetc(fp))!=EOF){
                x=c;//Needs to subtract by '0' to convert to a proper integer.
                sum=sum+x*power(x,i);//x in power should be replaced by 10
                i--;
        }
        printf("The value is %d", sum);
}

int power(int number, int exp){
        int i;
        int value=0;//Value must start at 1
        for(i=0;i<exp;i++){
                value=value*number;
        }

        return number;//should return value
}
```

8. (8 points) This question tests your understanding of multi-dimensional arrays. The following code is meant to take the "transpose" of a matrix—or switch the rows with the columns. For example, if we have the matrix:

```
| A B C |
| D E F |
| G H I |
```

Then the transpose would be:

```
| A D G |
| B E H |
| C F I |
```

Fill in the blanks to complete this program so that it takes the transpose of `array` and then prints it out.

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

int main() {
    int array[10][10] = {{0,1,2,3,4,5,6,7,8,9},
                         {0,1,2,3,4,5,6,7,8,9},
                         {0,1,2,3,4,5,6,7,8,9},
                         {0,1,2,3,4,5,6,7,8,9},
                         {0,1,2,3,4,5,6,7,8,9},
                         {0,1,2,3,4,5,6,7,8,9},
                         {0,1,2,3,4,5,6,7,8,9},
                         {0,1,2,3,4,5,6,7,8,9},
                         {0,1,2,3,4,5,6,7,8,9},
                         {0,1,2,3,4,5,6,7,8,9}};
    int transpose_array[____10____][____10____];
    int i,j;

    for (i=0;i<10;i++) {
        for (j=0;j<10;j++) {
            transpose_array[__i__][__j__] = array[__j__][__i__];
        }
    }

    for (i=0;i<10;i++) {
        for (j=0;j<10;j++) {
            printf("%d␣",transpose_array[i][j]);
        }
        printf(____"\n"____);
    }

    return 0;
}
```

Solution:

SOLUTION:

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

int main() {
    int array[10][10] = {{0,1,2,3,4,5,6,7,8,9},
                         {0,1,2,3,4,5,6,7,8,9},
                         {0,1,2,3,4,5,6,7,8,9},
                         {0,1,2,3,4,5,6,7,8,9},
                         {0,1,2,3,4,5,6,7,8,9},
                         {0,1,2,3,4,5,6,7,8,9},
                         {0,1,2,3,4,5,6,7,8,9},
                         {0,1,2,3,4,5,6,7,8,9},
                              {0,1,2,3,4,5,6,7,8,9},
                              {0,1,2,3,4,5,6,7,8,9}};
    int transpose_array[10][10];
    int i,j;

    for (i=0;i<10;i++) {
        for (j=0;j<10;j++) {
            transpose_array[i][j] = array[j][i];
        }
    }

    for (i=0;i<10;i++) {
        for (j=0;j<10;j++) {
            printf("%d ",transpose_array[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```