Université d'Ottawa
Faculté de genie

École d'ingénierie et de
technologies de l'information

uOttawa
L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

School of Information
Technology and Engineering

GNG 1106 Fall 2014 Final Exam (English)

- Closed-book
- 180 minutes
- The paper is out of 100 points, but you can get up to 108 points. That is, there are additional 8 bonus points.

Part 1: Code Reading Questions ($7 \times 9 = 63$ Points)

Question 1 *What does the following program print?*

```
#include <stdio.h>

struct wedget
{
  int x;
  int y;
};
void modifyWedget(struct wedget inputWedget, int newX, int newY)
{
    inputWedget.x=newX;
    inputWedget.y=newY;
}
void printWedget(struct wedget inputWedget)
{
    printf("the wedget has x=%d and y=%d\n", inputWedget.x, inputWedget.y);
}
int main()
{
    struct wedget myWedget;
    myWedget.x=1;
    myWedget.y=2;
    printWedget(myWedget);
    modifyWedget(myWedget, 10, 20);
    printWedget(myWedget);
    return 0;
}
```

Question 2 *What does the following program print?*

```
#include <stdio.h>

void doIt(int *x)
{
    printf("%d\n", *x);
    if (*x%2==0)
        *x=*x-5;
    else
        *x=*x-3;
}

int main()
{
    int a=19;
    while(a>0)
        doIt(&a);
    return 0;
}
```

Question 3 *What does the following program print?*

```
#include <stdio.h>
int main()
{
    int x=10;
    while(x<20)
    {
        printf("x is %d\n", x);
        if ((x/2)*2==x)
            x++;
        else
            x=x+3;
    }
    return 0;
}
```

Question 4 *What does the following program print?*

```
#include <stdio.h>

struct array
{
  int length;
  int *data;
};

void printArray(struct array theArray)
{
    int i;
    printf("printing array:\n");
    for (i=0;i<theArray.length;i++)
        printf("%d\t", theArray.data[i]);
    printf("done printing array\n");
}

int main()
{
    int A[5]={1, 2, 3, 4, 5};
    int i;
    struct array a;
    struct array b;
    a.length=5;
    a.data=A;
    b=a;
    for (i=0;i<a.length;i++)
        a.data[i]=a.data[i]+10;
    printArray(a);
    printArray(b);
    return 0;
}
```

Question 5 *The following code segments all intend to use variable* `x` *to specify a string "*`hello`*". Verify if they are correct.*

1. ```
   char x[256]="hello";
   ```

2. ```
   char x[256];
   x="hello";
   ```

3. ```
   char *x="hello";
   ```

4. ```
   char *x;
   x="hello";
   ```

5. ```
   char x[256]={'h', 'e', 'l', 'l', 'o'};
   ```

Question 6 *What does the following program print?*

```
#include <stdio.h>

int getIt()
{
    static int a=2;
    static int b=1;
    int out;
    out=a+b;
    a=b;
    b=out;
    return out;
}

int main()
{
    printf("%d\n", getIt());
    printf("%d\n", getIt()+getIt());
    printf("%d\n", getIt()+getIt()/4);
    return 0;
}
```

Question 7 *Determine if the program below is correct. If it is correct, explain what it prints; otherwise, explain why it is wrong.*

```
#include <stdio.h>
int main()
{
    int i;
    int a[5]={1, 2, 3, 4, 5};
    int b[5]={6, 7, 8, 9, 10};
    b=a;
    for (i=0;i<5;i++)
        printf("%d\n", b[i]);

    return 0;
}
```
```

Question 8 *Determine if the program below is correct. If it is correct, explain what it prints; otherwise, explain why it is wrong.*

```
#include <stdio.h>
int main()
{
    int i;
    int a[5]={1, 2, 3, 4, 5};
    int b[5]={6, 7, 8, 9, 10};
    *b=*a;
    for (i=0;i<5;i++)
        printf("%d\n", b[i]);

    return 0;
}
```

Question 9 *Determine if the program below is correct. If it is correct, explain what it does (by providing example input to the program as well as its corresponding output); otherwise, explain why it is wrong.*

```
#include <stdio.h>

int *getThreeNumbers()
{
    int i;
    int a[3];
    int *p;
    for (i=0;i<3;i++)
    {
        printf("enter an integer\n");
        scanf("%d", a+i);
    }
    p=a;
    return p;
}

int main()
{
    int *q;
    int i;
    q=getThreeNumbers();
    printf("the number you entered are:\n");
    for (i=0;i<3;i++)
        printf("%d\n", q[i]);
    return 0;
}
```

Part 2: Programming Questions ($15 \times 3 = 45$ points)

- **Four questions are given in this part, but you are to choose any three to solve. If you work on all four questions, make sure you specify which three are to be graded.**
- **In all programming questions, no global variables are allowed.**

Question 10 *Write a function that takes an array of* `int` *typed values as input such that when the function is called, the calling function gets the largest value and smallest value in the array. You are free to design the prototype of the function (i.e., free to design the input and output variables of the function). Also write a* `main` *function which calls this function to demonstrate its use.*

Question 11 *A structure* `studentRecord` *is defined as below, which is used to represent the record of a student in a course.*

```
struct studentRecord
{
    char lastName[100];
    char firstName[100];
    long studentId;
    double mark;
};
```

Implement a function with the following prototype

```
void writeRecordsToFile(struct studentRecord *records, int numOfRecords, char *filename);
```

The function write an array of `struct studentRecord`*-typed data to a specified file in ascending order of the student ID's. More specifically, the input variable* `records` *specifies the address of the input array; the input variable* `numOfRecords` *specifies the number of student records in the array; and* `filename` *specifies (the address of) the file name string of the file which will be created and written into. In addition, in the created file, each row contains one student record, and the ordering of the students records are in ascending order of the student ID's (namely, in ascending order of their* `studentId` *values).*

Question 12 (15 points) *Write a function with name* `getOddNumbers`*. The purpose of the function is to process an input array of* `int` *typed integers and return the list of odd numbers in the array. The function needs to be able to handle input arrays with arbitrary lengths. The return type of the function must be "*`int *`*", or pointer to* `int`*, namely that the function returns the address pointing to the memory storing the list of found odd numbers. Except for these requirements, you are free to design the function. You also need to write a testing program (namely a* `main` *function) to show that your design and implementation of this function are correct. Your testing program needs to do the following: first it asks the user to enter a positive integer N, then it generates a length-N array of random integers; then it calls your function* `getOddNumbers` *to get the list of odd numbers in this array, and finally it prints the list.*

Question 13 *Just like decimal numbers are represented using 10 different symbols (i.e.,* $\{0,1,2,3,4,5,6,7,8,9\}$*), hexadecimal numbers are represented using 16 different symbols. These 16 symbols are*

$$\{0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f\}.$$

The numerical (decimal) values of the 16 symbols are given in the table below.

Numerical Value	*Hexadecimal Symbol*
0	*0*
1	*1*
2	*2*
3	*3*
4	*4*
5	*5*
6	*6*
7	*7*
8	*8*
9	*9*
10	*a*
11	*b*
12	*c*
13	*d*
14	*e*
15	*f*

For any positive integer K*, an integer number* x *between* 0 *and* $16^K - 1$ *can be converted to a* K*-symbol hexadecimal representation. We explain this conversion rule by taking the example of* $K = 4$ *(and hence* x *is between* 0 *and* $65535 = 16^4 - 1$*).*

1. *Express* x *as* $x = y_3 \cdot 16^3 + y_2 \cdot 16^2 + y_1 \cdot 16 + y_0$*, where* y_0, y_1, y_2, y_3 *are all numbers in* $\{0, 1, \ldots, 15\}$*. Notice that for every integer* x *between* 0 *and* 65535*, such an expression is unique.*

2. *Convert the ordered tuple* (y_3, y_2, y_1, y_0) *symbol-by-symbol to hexadecimal symbols using the above table.*

- *Example 1:* $x = 15678$*. Express* x *as* $x = 3 \cdot 16^3 + 13 \cdot 16^2 + 3 \cdot 16 + 14$*. That is,* $y_3 = 3, y_2 = 13, y_1 = 3, y_0 = 14$*. The* (y_3, y_2, y_1, y_0) *tuple is then* $(3, 13, 3, 14)$*. Converting this tuple to hexadecimal symbols gives us string* **3d3e**

- *Example 2:* $x = 3258$*. Express* x *as* $x = 0 \cdot 16^3 + 12 \cdot 16^2 + 11 \cdot 16 + 10$*. That is,* $y_3 = 0, y_2 = 12, y_1 = 11, y_0 = 10$*. The* (y_3, y_2, y_1, y_0) *tuple is then* $(0, 12, 11, 10)$*. Converting this tuple to hexadecimal symbols gives us string* **0cba**

Clearly the above rule extends to arbitrary positive integer K*. That is, to convert an integer* x *between* 0 *and* $16^K - 1$ *to a* K*-symbol representation, we only need to express* x *as*

$$x = y_{K-1} \cdot 16^{K-1} + y_{K-2} \cdot 16^{K-2} + \ldots + y_2 \cdot 16^2 + y_1 \cdot 16^1 + y_0.$$

Then we convert the tuple $(y_{K-1}, y_{K-2}, \ldots, y_2, y_1, y_0)$ *to hexadecimal symbols according to the above table.*

The following program is meant to do the following: it asks the user to enter a non-negative integer x *and a positive integer* K*; if* x *is between* 0 *and* $16^K - 1$*, it prints the* K*-symbol hexadecimal representation of the integer, otherwise it prints message "*`x is out of range`*". You are required to complete the program by designing and implementing the function* `printHex` *called in the* `main` *function.*

```
#include <stdio.h>
/*

Fill in the code here for function printHex

*/
int main()
{
        long x;
        int K;
        printf("enter non-negative integer x\n");
        scanf("%ld", &x);
        printf("enter positive integer K\n");
        scanf("%d", &K);
        printHex(x, K);
        return 0;
}
```