CSCE 206 Fall2019 Lab: Assignment #5

Submission Deadline: 23:59, Dec 02, 2019, Monday.

1. Follow the submission guideline to submit the assignment through eCampus.
2. Add comments to your code, including your name, UIN and the class section you are in with the block comments to the head of your code file.
3. **!!!Name your folder with UIN and Full Name then zip to submit!!!**

Question 1. **Hex Calculator** (40 points)

Write your code to implement +, -, *, /, four operations between two hex numbers. The program is required to accept a single C string formula as input (10 points) and then properly separate two hex numbers and an operator from the input (10 points) and finally perform the operation according to the operator and return a correct decimal answer on terminal/console (10 points). Name your program file **Lab5_q1_code.c**.

Requirements are following:

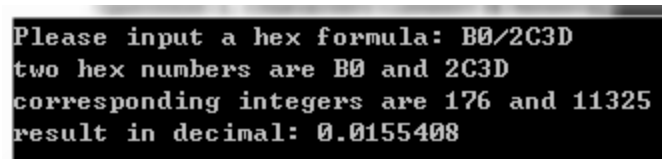
1. **Not allowed** to use any other libraries as your head files except <stdio.h> or <math.h>. Write the conversion function by yourself.
2. Use **scanf** function to accept input and input type must be stored as a character array variable.
3. Probably you will pick some code from your previous Lab 4.

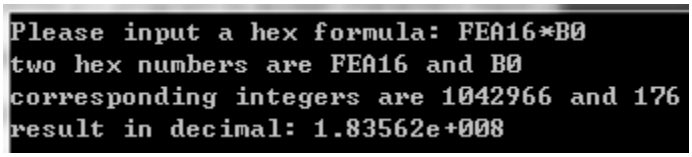
Sample inputs/outputs:

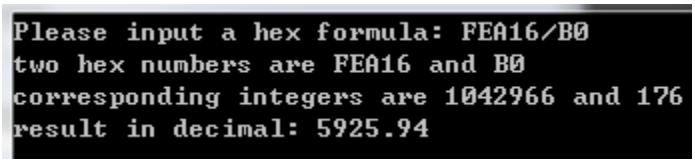
```
Please input a hex formula: B0+2C3D
two hex numbers are B0 and 2C3D
corresponding integers are 176 and 11325
result in decimal: 11501
```

```
Please input a hex formula: B0-2C3D
two hex numbers are B0 and 2C3D
corresponding integers are 176 and 11325
result in decimal: -11149
```

```
Please input a hex formula: B0*2C3D
two hex numbers are B0 and 2C3D
corresponding integers are 176 and 11325
result in decimal: 1.9932e+006
```







Hint:

1. Go over Ch4, Ch5, Ch8, Ch9 slides. Understand how a pointer works on an array and how to pass a C string to a function (call by reference or call by pointer);
2. Go over Ch3 & Ch6 slides. Understand how to use switch statement and how to handle datatype conversions (integer to float/double and float/double to integer);
3. Borrow your code from Question 2, Lab4 and merge it as a hex2int or hex2double function so that it could be called inside main;
4. Properly separate those numbers and operator from the input C string.

Question 2. Counting Game (60 points)

Given a group of N players sitting in a circle on N chairs that have been labelled in order by $1{\sim}N$. The player sitting on the k^{th} $(k \leq N)$ starts to clockwise number off by M. Each time the player saying M will leave the seat and the rest players continue and until the last player left. That is the person sitting on the k^{th} chair saying 1, person sitting on the $(k+1)^{th}$ chair saying 2, the person sitting on the $(k+2)^{th}$ chair saying 3, so and so forth, until the person sitting on the $(k+M-1)^{th}$ chair saying M. Then the person sitting on the $(k+M-1)^{th}$ leaves the chair and rest $(N-1)$ persons continue this game. The last player being left on chair will be the winner.

For example, if $N=5, k=2, M=3$, will have (* denotes a player),

Chair label	1	2	3	4	5
Before	*	*(saying 1)	*	*	*
1st round	*	*	*		*(saying 1)
2nd round	*		*(saying 1)		*
3rd round			*(saying 1)		*
4th round					*(winner)

if $N=5, k=5, M=4$, will have (* denotes a player),

Chair label	1	2	3	4	5
Before	*	*	*	*	*(saying 1)

1st round	*	*		*(saying 1)	*
2nd round	*			*(saying 1)	*
3rd round	*				*(saying 1)
4th round					*(Winner)

Write a code to accept three numbers as input for N, k, M, respectively and play this game and find who will be the winner. Also print the real-time chessboard (chairboard) in each round (30 points). Name your program file **Lab5_q2_code.c**.

Requirements are following:

1. **Not allowed** to use any other libraries as your head files except <stdio.h> or <math.h> or <malloc.h>.

Sample inputs/outputs:

```
Input the quantity of players (N), the starting player (k), and the M:
5, 2, 3
Iteration: 1
removing the player sitting on the chair label: 4
remaining players are:
a player sitting on the chair label: 5
a player sitting on the chair label: 1
a player sitting on the chair label: 2
a player sitting on the chair label: 3
Total remaining players: 4

Iteration: 2
removing the player sitting on the chair label: 2
remaining players are:
a player sitting on the chair label: 3
a player sitting on the chair label: 5
a player sitting on the chair label: 1
Total remaining players: 3

Iteration: 3
removing the player sitting on the chair label: 1
remaining players are:
a player sitting on the chair label: 3
a player sitting on the chair label: 5
Total remaining players: 2

Iteration: 4
removing the player sitting on the chair label: 3
remaining players are:
a player sitting on the chair label: 5
Total remaining players: 1

The winner player is sitting on the chair label: 5
```

```
Input the quantity of players (N), the starting player (k), and the M:
5, 5, 4
Iteration: 1
removing the player sitting on the chair label: 3
remaining players are:
a player sitting on the chair label: 4
a player sitting on the chair label: 5
a player sitting on the chair label: 1
a player sitting on the chair label: 2
Total remaining players: 4

Iteration: 2
removing the player sitting on the chair label: 2
remaining players are:
a player sitting on the chair label: 4
a player sitting on the chair label: 5
a player sitting on the chair label: 1
Total remaining players: 3

Iteration: 3
removing the player sitting on the chair label: 4
remaining players are:
a player sitting on the chair label: 5
a player sitting on the chair label: 1
Total remaining players: 2

Iteration: 4
removing the player sitting on the chair label: 1
remaining players are:
a player sitting on the chair label: 5
Total remaining players: 1

The winner player is sitting on the chair label: 5
```

More test cases:

$N = 10, k = 2, M = 11$ => Winner: 8

```
a player sitting on the chair label: 8
Total remaining players: 5

Iteration: 6
removing the player sitting on the chair label: 10
remaining players are:
a player sitting on the chair label: 3
a player sitting on the chair label: 5
a player sitting on the chair label: 6
a player sitting on the chair label: 8
Total remaining players: 4

Iteration: 7
removing the player sitting on the chair label: 6
remaining players are:
a player sitting on the chair label: 8
a player sitting on the chair label: 3
a player sitting on the chair label: 5
Total remaining players: 3

Iteration: 8
removing the player sitting on the chair label: 3
remaining players are:
a player sitting on the chair label: 5
a player sitting on the chair label: 8
Total remaining players: 2

Iteration: 9
removing the player sitting on the chair label: 5
remaining players are:
a player sitting on the chair label: 8
Total remaining players: 1

The winner player is sitting on the chair label: 8
```

$N = 41, k = 1, M = 3$ => Winner: 31

```
a player sitting on the chair label: 31
a player sitting on the chair label: 35
Total remaining players: 5

Iteration: 37
removing the player sitting on the chair label: 22
remaining players are:
a player sitting on the chair label: 31
a player sitting on the chair label: 35
a player sitting on the chair label: 4
a player sitting on the chair label: 16
Total remaining players: 4

Iteration: 38
removing the player sitting on the chair label: 4
remaining players are:
a player sitting on the chair label: 16
a player sitting on the chair label: 31
a player sitting on the chair label: 35
Total remaining players: 3

Iteration: 39
removing the player sitting on the chair label: 35
remaining players are:
a player sitting on the chair label: 16
a player sitting on the chair label: 31
Total remaining players: 2

Iteration: 40
removing the player sitting on the chair label: 16
remaining players are:
a player sitting on the chair label: 31
Total remaining players: 1

The winner player is sitting on the chair label: 31
```

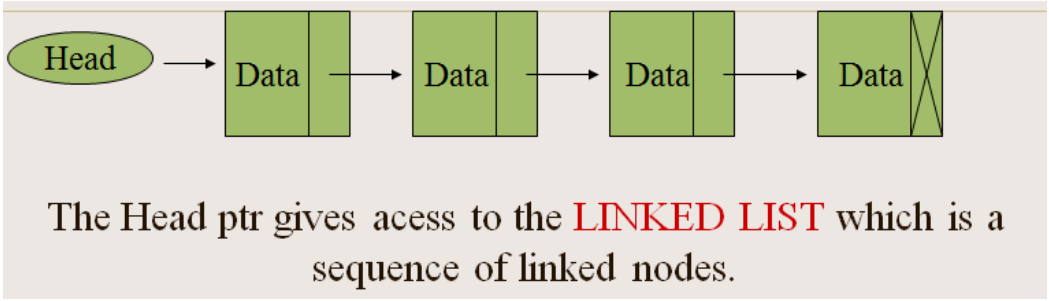
Hint:

1. There are two ways to go:
   1) Using a single-dimension array;
   2) Using the struct type in C.

   Basically, (1) will refer to Ch9 slide talking about Array and (2) will refer to Ch12 slide talking about C structure. However, both will probably refer to Ch8 slide talking about Pointer.

   If choose (1), probably will consider two things: a. How to denote a player in an array who leaves his or her seat? b. How to loop on an array (or in other words, which operator in C could result in a loop? Check the operators slide.)?

   If choose (2), probably will use looped linked list (linked list cycle, check below, it is a linked list connecting its tail to the head) and primitive list functions (insert a node, delete a node). Be careful when you delete a node. You will have to use *free(node)* in <malloc.h> so that the memory could be properly recycled. Tutorial is here.
2. Below is a linked list captured from the C structure slide.



   The only difference between the linked list cycle and the linked list is the linked list cycle will have its tail and head connected while the linked list has its nulled tail only.