Assignment 4

Due Sunday, August 4th, 2013

This assignment is worth 6% of your grade.

Submit your code on cuLearn in ONE file called a4.c Your code (that is a4.c) must compile, otherwise the assignment will be graded with mark 0. For that reason, if you run out of time and/or one of your answers contains code that does not compile, then comment out that section of code. You can use the following libraries: `stdio.h` and `string.h`

To solve this assignment you will need to know arrays, strings and a bit about FILES. You can start by coding function `bubble_sort` since you have seen arrays already. You will hear about strings on Tuesday (see chapter 8).

You should write a program that does the following:
****

1) Opens a file called “input.txt”. I will provide a sample of this file on cuLearn. You should put it in the same folder on your computer where a4.c resides. This file contains a list of even integers in the range [20, 78] but written with letters. For example the file may look like this:

```
twentyeight
thirty
fortytwo
thirty
fiftysix
fiftyfour
seventyfour
sixtyfour
twenty
```

2) You program should then read the numbers from this file, one by one until it reaches end of file (EOF).

Each time it reads a number from the file, the program should
a) store it into a string
b) pass this string as an input to a function called `convert` which will return an integer equal to that number

All of these integers should be stored in an integer array called `nums`

3) Then you program should call a function called `bubble_sort` which will sort the elements of the array `nums` from the smallest to the largest. To sort the numbers your program should use *bubble sort* algorithm explained in problem 13 of Chapter 7 (see page 450). Alternatively you can find wikipedia article on bubble sort.

When this `bubble_sort` function returns, the main should print sorted integers without *duplicates*.
****

You can test your program on the file input.txt that I provided. However, the TA will test your program on a different input.txt file. Your program should thus work for any file as long as that file is called `input.txt` and contains at most 100 numbers in the format specified in 1) above.

To learn how to open and close a file in C, read pages 318, 319, 629 and 630 from your book. Also see the code I proved below. It basically has everything you need to know about files (for this assignment). You should use it as a skeleton of your program.

The function `convert` should use as few “if” (or “elseif”) comparisons as possible. Surely you do not want to use 30 `strcmp` calls to do these comparions. Find your own, better way.

Example: Here is what your program should print if the file `input.txt` had following content:

```
twentyeight
thirty
fortytwo
thirty
fiftysix
fiftyfour
seventyfour
sixtyfour
twenty
fiftysix
thirty
```

Before the call to `bubble_sort` your array nums should have the following numbers at the beginning of the array
28, 30, 42, 30, 56, 54, 74, 64, 20, 56, 30

After the call to the `bubble_sort` your array nums should have the following numbers at the beginning of the array
20, 28, 30, 30, 30, 42, 54, 56, 56, 64, 74

It should then print
20, 28, 30, 42, 54, 56, 64, 74

You should use the following skeleton for your program

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

/* here go prototypes of your two functions convert and bubble_sort */

int main(void)
{
  char str_num[20];
  int nums[100];
  FILE *inp;

  inp=fopen("input.txt","r");

  /* if fopen fails to open a file it returns a NULL pointer */
  if(inp==NULL){
    printf("\n Cannot open file input.txt \n");
    return(-1); /* the program terminates immediately */
  }

   while(fscanf(inp,"%s", str_num)!=EOF){
        /*you can use this command to see what fscanf just read */
        /* printf("\n%s\n", str_num); */

    /* here goes the call to your function convert as, well as,
       storing of the result it returns into the array nums */

  }

  /* here goes the call to the function bubble_sort */

  /* here goes printing of the sorted elements of the array without
     printing duplicate numbers */

  fclose(inp);
  return(0);
}

/* here go your two functions */
```