COP 3223 Sec 2: Fall'11 C Programming Prac Test 4 (?? points)

1. (7 points) Write a program that lists all the capital letters in reverse alphabetical order (Z to A). Make sure you use a loop, DO NOT USE AN ARRAY, and keep your program to under 12 lines of code.

2. (?? points)
   Do question 1, but do it within a void function that takes two ints as pass-by-value parameters. These two ints give the beginning and the end of the range of positions within the alphabet sequence (A is zero). Also write a main program to call this function.

3. (10 points) Write a program and a void function. The void function will take two ints as pass-by-value parameters. These two ints give the start and the end (the start guaranteed to be smaller than the end) of the range of positions within the alphabet sequence (A is zero, B is one, etc.) The function must then print the start capital letter and subsequently every capital letter in the ascending alphabetical order till including the end letter. Make sure the function uses a FOR loop. You may use putchar() or printf.

   Also write all the headings (include's, DEFINES, etc.) and the main program to call this function. Before the call, the main should read in two (guaranteed uppercase letters) chars using `scanf("%c%c",&startchar,&endchar)`, and then pass the integer positions of these two uppercase letters to your function. Do not prompt for input, do not error-check, and do not bother with reading in the dummy char at the end of the line.

   E.g., If the user types in "CJ", the ints passed to the function should be 2 and 9, and the printed output will be "CDEFGHIJ". If the user types in "HP", the ints sent to the function should be 7 and 15, and the printed output will be "HIJKLMNOP".

   You may find this scale helpful in understanding the examples.

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
A B C D E F G H I J  K  L  M  N  O  P  Q  R  S  T  U  V  W  X  Y  Z
```

4. (?? points)
   Write a function to take two struct records (fields are SSN, and wages earned as floats) as pass-by-value, and if the SSN's agree, the 2 wages should be added and passed back through a third (pass-by-reference) parameter of type struct record. If the SSN's agree, the integer 1 should be returned through the function's name. If they do not agree, the integer zero should be returned and the third parameter should not be tampered with.

5. (10 points)
   Suppose a struct type named Payment has been defined with two fields: an int SSN, and a float fine (which is a traffic ticket fine). `struct Payment {int SSN; float fine};`

   Write **ONLY** an int function named IncreaseFine to take two parameters of this type as pass-by-value, and if the SSN's agree, the 2 fines should be added and a quantity that is double their sum should be passed back through a third (pass-by-reference) parameter of type Payment (also, copy the SSN from the first

parameter into the third). Additionally, if the SSN's agree, the integer 1 should be returned through the function's name. If they do not agree, the integer zero should be returned and the third parameter should not be tampered with.

6. (10 points) Suppose you have this structure:

```
struct gas {
          float distance;
          float gals;
          float mpg;
      };
```

A. Write ONLY the function that takes a `struct gas` argument. Assume that the passed structure contains the `distance` and `gals` information. The function should calculate the correct value for the `mpg` field and return (through the function name) the now completed structure.

B. Write only the function that takes the address of a `struct gas` argument. Assume that the passed structure contains the `distance` and `gals` information. The function should calculate the correct value for the `mpg` field and assign it to the appropriate field, so that when the function has ended, the now completed struct will be available to the caller.

7. (10 points)
Write a **complete** program that reads in from the first line an integer n, and thereafter on n subsequent lines six things per line (an int SSN and five float values day1wages, day2wages, ... day5wages). All this is from a file. You must prompt for the filename as a string that is certain to be less than 30 chars long. Open the file, etc., read everything, and keep a running total of all the wages (there will be n times 5 values going into the total); then when reading is complete, prompt for an output filename string, open it and write to it the total value of all the wages paid by the system. Remember to close the files.

8. (16 points)
A. Assume the following given declarations:

```
        struct TuitionOwed {int SSN; float tuition};
and
        struct DateToday {int month; int day; int year};
```

Write ONLY the function UpdateTuition that takes in one struct TuitionOwed and one struct DateToday (both parameters are pass-by-value), and returns a new struct TuitionOwed via the function's name. The returned struct will have to get the SSN from parameter one, and will either increase the tuition by 10% or 0% depending on whether the DateToday shows a date after August 31 or not (this is a lateness penalty). Hint: You only need to check: if month is greater than 8, then increase by 10%, else simply copy the incoming tuition to the outgoing tuition.
B. Write ONLY the same function as named above (and doing the same job as the one above), but this time include a third parameter as pass-by-reference (it should be a pointer to a struct TuitionOwed); this third parameter will contain the updated tuition and the SSN. The first two parameters are pass-by-value, as in question above. This time, make the function be void.

9. (15 points)
Suppose you have this structure:

```
struct complex_numb{
          float real;
          float imaginary;
      };
```

A. Write ONLY the function Add_Complex that takes two complex numbers called C1 and C2, and returns (via the function's name) a complex number that represents the sum of the two input complex numbers (please recall from high-school algebra that this is done by adding the two real components and the two imaginary components separately).
B. Write ONLY the function Copy_Complex (of type void) that takes two arguments: a first (the parameter on the left) complex number that is pass-by-reference, and a second (the parameter on the right) complex number that is pass-by-value. The values in the second complex number should be copied into the first complex number, so that this is like an "assignment" statement for complex numbers.
So, a call to this might look like `Copy_Complex(&c1 , c2);`

10. (?? points)
Write a function to take three params: an int** array named PIC1 and two ints named numRows and numCols. A fourth int parameter is named Value1. Combine the above with each requirement below to get a new problem for you to solve.

a) The int function should loop through the rows and columns, and return thr count of the number of elements in PIC1 whose value is Value1.

b) The void function should loop through the rows and columns and if the lement in PIC1 is greater than Value1, the element should be set to 255, otherwsie set it to zero.

c) The void function should add a disk (of brightness 50) of radius Value1 centered at numRows/2 and numCols/2.

d) The void function should draw the horizontal and vertical axis (of brightness 255) centered at position [Value1, Value1].

e) The void function should draw a diagonal line of brightness Value1 through the pixel centered at numRows/2 and numCols/2 (Assume that numRows is same as numCols).

11. (?? points)
Write a program that reads in from the first line an integer n, and thereafter on n subsequent lines two things per line (an int SSN and a float wages-earned). All this is from a file. You must prompt for the filename as a string that is certain to be less than 30 chars long. Open the file, etc., read everything, and keep a running total of all the wages earned, then when reading is complete, prompt for an output filename string, open it and write to it the average value of all the earnings.

12. (?? points) Write **ONLY** a void function to take three parameters: an `int**` array named PIC1 and two ints named numRows and numCols. Assume numRows and numCols are always even numbers (not odd). The function should copy the right half of the picture into the space currently occupied by the left half, but also turn it upside down, as in the figure.

Desired output for the face06.pgm picture

13. (10 points) What is the output of this program caused by the first and sixth call to Print_Array?

```
#include <stdio.h>
#include <stdlib.h>
#define         SIZE    10
void Print_Array(int values[], int length);
void swap(int values[], int i, int j);
void Move_Max(int values[], int max_index);
void Simple_Sort(int values[], int length);
int main() {
  int my_vals[SIZE] = {84, 91, 17, 55, 42, 36, 55, 29, 74, 67};
  printf("original array : ");
  Print_Array(my_vals, SIZE);
  Simple_Sort(my_vals, SIZE);
}
void Simple_Sort(int values[], int length)
{
  int i;
  for (i=length-1; i> 0; i--)
  {
    Move_Max(values, i);
    Print_Array(values, SIZE);
  }
}
void Move_Max(int values[], int max_index)
{
  int max, i, maxi;
  max = values[0];
  maxi = 0;
  for (i=1; i<=max_index; i++)
  {
    if (max < values[i])
    {
      max = values[i];
      maxi = i;
    }
  }
  swap(values, maxi, max_index);
}
void swap(int values[], int i, int j)
{
  int temp;
  temp = values[i];
  values[i] = values[j];
  values[j] = temp;
}
void Print_Array(int values[], int length)
{
  int i;
  for (i=0; i<length; i++)
    printf("%d ", values[i]);
  printf("\n");
}
```