C Assignment 3

Name: Diego Trazzi
Student ID: 300338937

01 (4 marks) See the following two statements.

```
int a[3] = {11, 22, 33};
int *pa = a;
```

Give the values of the following expressions.

```
*a = 11
*(a+2) = 33
*pa = 11
pa[1] =22
```

02 (6 marks) See the following two statements.

```
int m[4][4] = {{1,3,5,7}, {11,33,55,77}, {2,4,6,8}, {22,44,66,88}};
int (*parr)[4] = m; // pointer to an array, because [] has precedence on *
```

Give the values of the following expressions.

```
**m = 1
*(*m+2) = 5 // dereference m (1), plus 2 => 3, then dereferenced => 5
*(*(m+1)+1) =  33
*(m[1]+2) = 55
(*(m+2))[3] = 8
(*(parr+3))[2] = 66
```

03 (6 marks) Suppose you are working on a 32-bit machine where the size of an int is FOUR bytes, the size of a char is ONE byte and the size of pointers is FOUR bytes. See the following statements.

```
char *pa[] = {"12", "34", "56"}; // array of pointers to char
int m[2][3] = {{1, 2, 3}, {4, 5, 6}}; // 2d array
int (*ppm)[2][3] = &m; // pointer to a 2d array
```

Give the values of the following expressions.

```
sizeof(pa)    = an array of size 3 of pointers to char -> 4*3 = 12 bytes
sizeof(*pa)   = a pointer to the first element of pa[] -> 4 bytes
sizeof(**pa) = is '1', a single char -> 1 byte
sizeof(ppm)   = (*ppm)[2][3] is a pointer to a 2D array, so pointer ppm -> 4 bytes
sizeof(*ppm) = (*ppm)[2][3] is a pointer to a 2D array, so 4x3 -> 24 bytes 24
sizeof(**ppm)= (*ppm)[2][3] is a pointer to a 2D array, so 3 int -> 12 byte
```

04 (6 marks) Declare p.

```
int arr[];
int *arr[] -> array of pointers
int (*arr)[] -> pointer to array
```

p is a 5-element array of pointers to char.

```
char *p[5]
```

p is a pointer to a 10-element char array.

```
char (*p)[10]
```

p is a function that takes an int argument and returns a pointer to char.

```
char *(p(int))
```

p is a function that takes a char array and returns a pointer to int.

```
int *(p(char [])) // would be the same if was: int *p(char [])
```

p is a pointer to a function that takes two int arguments and returns a pointer to an int.

```
int *((*p)(int, int)) // what about: int *(*p)(int int) ?
```

p is a function that takes no arguments and returns a pointer to a function that takes an int argument and returns a pointer to a 10-element int array.

```
int (*(*(p(void))(int)))[10]
```

Programming Questions

05 (7 marks) Complete the following program that prompts user to enter two dates and then indicates which date comes earlier on the calendar.

```
/*
 * q5.c
 *
 *  Created on: 19/05/2015
 *      Author: diego
 */

#include <stdio.h>
/* Note: Program assumes years are in the same century. */

struct date {
 int month, day, year;
};

int compare_dates(struct date d1, struct date d2);
void put_date(struct date d);
int check_date(struct date d);

int main(void) {
 struct date d1, d2;

 printf("Enter first date (mm/dd/yy): ");
 /* for you to complete */
 scanf("%d/%d/%d", &d1.month, &d1.day, &d1.year);

 printf("Enter second date (mm/dd/yy): ");
 /* for you to complete */
 scanf("%d/%d/%d", &d2.month, &d2.day, &d2.year);

 // chceck dates
 if (check_date(d1) || check_date(d2) < 0 ){
```

```
 printf("Invalide dates.");
      return -1;
 }

 int a = compare_dates(d1, d2);

 if (a < 0) {
      put_date(d1);
      printf(" is earlier than ");
      put_date(d2);
      printf("\n");
 } else if (a > 0) {
      put_date(d2);
      printf(" is earlier than ");
      put_date(d1);
      printf("\n");
 } else {
      printf("Dates are equal !");
 }

 return 0;
}

int check_date(struct date d){
 if (d.year < 0 || d.year > 99) {
      return -1;
 }
 if (d.month < 1 || d.month > 12){
      return -1;
 }
 if (d.day < 1 || d.day > 31){
      return -1;
 }
 return 0;
}

int compare_dates(struct date d1, struct date d2) {
 if (d2.year < d1.year) {
      return 1;
 } else if (d2.year > d1.year) {
      return -1;
 } else { // same year
      if (d2.month < d1.month) {
           return 1;
      } else if (d2.month > d1.month) {
           return -1;
      } else { // same month
           if (d2.day < d1.day) {
                return 1;
           } else if (d2.day > d1.day) {
                return -1;
           } else { // same day !!
                return 0;
           }
      }
 }
}
```

```
    void put_date(struct date d) {
     printf("%d/%d/%.2d", d.month, d.day, d.year);
    }
```

06 **(5 marks) Rewrite the program in Question 5, using the following function prototype.**

```
int compare_dates(struct date *, struct date *); void put_date(struct date *);
```

```
    /*
     * q6.c
     *
     *  Created on: 19/05/2015
     *      Author: diego
     */

    #include <stdio.h>
    /* Note: Program assumes years are in the same century. */

    struct date { int month, day, year; };

    int compare_dates(struct date *, struct date *);
    void put_date(struct date *);
    int check_date(struct date d);


    int main(void) {
     struct date d1, d2;

     printf("Enter first date (mm/dd/yy): ");
     /* for you to complete */
     scanf("%d/%d/%d", &d1.month, &d1.day, &d1.year);
     //    printf("%d/%d/%d\n",d1.month, d1.day, d1.year);

     printf("Enter second date (mm/dd/yy): ");
     /* for you to complete */
     scanf("%d/%d/%d", &d2.month, &d2.day, &d2.year);

     // chceck dates
          if (check_date(d1) || check_date(d2) < 0 ){
          printf("Invalide dates.");
               return -1;
          }

     int a = compare_dates(&d1, &d2);

     if (a < 0) {
          put_date(&d1);
          printf(" is earlier than ");
          put_date(&d2);
          printf("\n");
     } else if (a > 0) {
          put_date(&d2);
          printf(" is earlier than ");
          put_date(&d1);
          printf("\n");
     } else {
        printf("Dates are equal !");
    }

    return 0;
}

int check_date(struct date d){
    if (d.year < 0 || d.year > 99) {
        return -1;
    }
    if (d.month < 1 || d.month > 12){
        return -1;
    }
    if (d.day < 1 || d.day > 31){
        return -1;
    }
    return 0;
}

int compare_dates(struct date *d1, struct date *d2) {
    if (d2->year < d1->year) {
        return 1;
    } else if (d2->year > d1->year){
        return -1;
    } else { // same year
        if (d2->month < d1->month){
            return 1;
        } else if (d2->month > d1->month) {
            return -1;
        } else { // same month
            if (d2->day < d1->day){
                return 1;
            } else if (d2->day > d1->day){
                return -1;
            } else { // same day !!
                return 0;
            }
        }
    }
}

void put_date(struct date *d) {
    printf("%d/%d/%.2d", d->month, d->day, d->year);
}
```

07 (5 marks) Write function swap_ptr which swaps the values between ptrp and ptrq. You also need to provide the function prototype for swap_ptr.

```
int swap_ptr (int *, int *);

int main(void){
    int a = 3;
    int b = 4;
    printf("Before swap a: %d b: %d\n",a, b);
    swap_ptr(&a, &b);
}
```

```
    int swap_ptr( int *ptrp, int *ptrq){
     int *tmp = ptrp;
     ptrp = ptrq;
     ptrq = tmp;
     printf("After swap a: %d b: %d\n",*ptrp, *ptrq);
     return 0;
    }
}
```

08 (11 marks) Complete the following program. The program should provide user a text menu with 5 options (see below). Options 0~3 are implemented by functions. That is, making choice of these four options will call the corresponding functions. In this program, you need to use arrays of pointers to functions and call these functions using pointers.

Enter a choice:
0 Print the array of grades
1 Find the minimum grade
2 Find the maximum grade
3 Print the average on all tests for each student
4 End program

```
    #include <stdio.h>
    #define STUDENTS 3
    #define EXAMS 4

    /* function prototypes */
    void minimum(int grades[][EXAMS], int pupils, int tests);
    void maximum(int grades[][EXAMS], int pupils, int tests);
    void average(int grades[][EXAMS], int pupils, int tests);
    void printArray(int grades[][EXAMS], int pupils, int tests);
    void printMenu(void);

    int main(void) {

     /* pointer to a function that takes as parameters a two-dimensional array and two
    integer values */
     void (*processGrades[4])(int[][EXAMS], int, int) = { printArray, minimum, maximum,
    average };
     // an array of pointers to a function which takes a 2D array of size X and EXAMS
    which takes int
     // I want to declare an array of pointers to functions instead of array of
    functions, so I can call the functions

     int choice = 0; /* menu choice */

     /* array of student grades */
     int studentGrades[STUDENTS][EXAMS] = { { 77, 68, 86, 73 }, { 96, 87, 89, 78 }, {
    70, 90, 86, 81 } };
     // a 2D array of int

     /* loop while user does not choose option 4 */
     while (choice != 4) {
          /* display menu and read user's choice */
          do {
               printMenu();
               scanf("%d", &choice);
          } while (choice < 0 || choice > 4); /* end do...while *//* pass choice into
    the array */

          /* for you to complete */
          processGrades[choice](studentGrades, STUDENTS, EXAMS);

     } /* end while */
     return 0; /* indicate successful termination */
    } /* end main */

    /* search for the minimum value */
    void minimum(int grades[][EXAMS], int pupils, int tests) {

     int i; /* loop counter */
     int j; /* loop counter */
     int lowGrade = 100; /* set lowGrade to highest possible score */

     /* for you to complete */
     for (i = 0; i<3; i++){
               for (j=0; j<4; j++){
                    if (grades[i][j] < lowGrade){
                         lowGrade = grades[i][j];
                    }
               }
          }
     printf("\n\tThe lowest grade is %d\n", lowGrade);
    } /* end function minimum */

    /* search for maximum value */
    void maximum(int grades[][EXAMS], int pupils, int tests) {

     int i; /* loop counter */
     int j; /* loop counter */
     int highGrade = 0; /* set highGrade to lowest possible score */

     /* for you to complete */
     for (i = 0; i<3; i++){
                    for (j=0; j<4; j++){
                         if (grades[i][j] > highGrade){
                              highGrade = grades[i][j];
                         }
                    }
               }

     printf("\n\tThe highest grade is %d\n", highGrade);
    } /* end function maximum */

    /* calculate average */
```

```c
void average(int grades[][EXAMS], int pupils, int tests) {

 int i; /* loop counter */
 int j; /* loop counter */
 int total = 0; /* sum of all grades */
 printf("\n");

 /* for you to complete */
 for (i = 0; i<3; i++){
           for (j=0; j<4; j++){
                total += grades[i][j];
                // printf("%d\n",total);
           }
      }
 printf("\n\tThe average grade is %d\n", total/(STUDENTS*EXAMS));

} /* end function average */

/* print the contents of the array */
void printArray(int grades[][EXAMS], int pupils, int tests) {

 int i; /* loop counter */
 int j; /* loop counter */
//     printf("\n\t [0] [1] [2] [3]\n");

 /* for you to complete */
 for (i = 0; i<3; i++){
      printf("[%d]\t",i);
      for (j=0; j<4; j++){
           printf("%d\t",grades[i][j]);
      }
      printf("\n");
 }

 printf("\n");
} /* end function printArray */

/* display the menu */
void printMenu(void) {

 printf(
           "\n\tEnter a choice:\n"
           "\t 0 Print the array of grades\n"
           "\t 1 Find the minimum grade\n"
           "\t 2 Find the maximum grade\n"
           "\t 3 Print the average on all"
           " tests for each student\n"
           "\t 4 End program\n"
           "\t? ");
} /* end function printMenu */
```