Anyone who stops learning is old, whether at twenty or eighty. Anyone who keeps learning stays young. The greatest thing in life is to keep your mind young.
~ *Henry Ford*

CS1010 Programming Methodology

Week 13: Structures, Files and Revision (Answers)

To DLs:

- Questions 1 and 2 are basic and should be discussed.
- You need not discussed all of questions 3, 4, and 5. You may choose to do one or two of them. You may want to inform students beforehand what questions you will be discussing.

To students:

Your DL may not cover all the questions here.

I. Structures

1. [CS1010 AY2010/2011 Semester 1 Exam, Q2a]

   Write down the output of the following program.

Download source code **Week13_Q1.c** from cs1010 account

```
#include <stdio.h>

typedef struct
{
    int i, a[4];
} mystruct_t;

int main(void)
{
    mystruct_t s, t;
    s.i = 5;
    s.a[3] = 10;
    t = s;
    printf("%d %d\n", t.i, t.a[3]);
    return 0;
}
```

Answer:

5 10

2. [CS1010 AY2011/2012 Semester 1 Exam, Q2c]

   Write down the output of the following program.

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

typedef struct
```

```
{
    char name[10];
    int age;
} person;

void func1(person *, char [], int);
void func2(person []);
void func3(person);

int main(void)
{
    person data[] = {{"Zhou", 25}, {"Tamil", 22},
                     {"Potter", 33} };
    func1(&data[0], "Ismail", 15);
    printf("%s %d\n", data[0].name, data[0].age);
    func2(data);
    printf("%s %d\n", data[1].name, data[1].age);
    func3(data[2]);
    printf("%s %d\n", data[2].name, data[2].age);
    return 0;
}

void func1(person *ptr, char name[10], int age)
{
    strcpy(ptr->name, name);
    ptr->age = age;
}

void func2(person per[])
{
    int i;
    for (i=0; i<3; i++)
        per[i].age++;
}

void func3(person per)
{
    strcpy(per.name, "Ace");
    per.age--;
}
```

Download source code **Week13_Q2.c** from cs1010 account

Answer:

Ismail 15
Tamil 23
Potter 34

3. Write a program **Week13_Q3.c** to read in an integer (assumed greater than 1) indicating the number of tiles, followed by the tiles' data (length, width and price per square metre). A structure called **tile_t** should be created and the tiles' data stored in an array of such structure. The program then computes and outputs the difference in cost between the cheapest tile and the most expensive tile.

   (Actually, to get the answer there is no need to store the data in an array. This is done just for exercise.)

   The length and width are integers in meters, while the price is of type **float**. You may assume that there are at least 2 tiles and at most 20 tiles.

   You should write a modular program with the following functions:

```
// To read tiles' data into array tiles
// *size_p contains the number of tiles to be entered by user
void scan_tiles(tile_t tiles[], int *size_p);

// Return the difference in cost between cheapest
// tile and most expensive tile in the array tiles
float difference(tile_t tiles[], int numTiles);
```

   A sample run is shown below.

```
Enter number of tiles: 5
Enter data for 5 tiles:
5 8 0.20
3 5 0.18
6 10 0.31
4 6 0.27
2 4 0.38
Largest difference = $15.90
```

Download skeleton **Week13_Q3.c** from cs1010 account

Answer: See **Week13_Q3.c**

This is a straight-forward question to get students acquainted with the syntax on structures. Please go through the student's programs and make sure that the class understand how to use structures.

4. [Past-year's CS1101 exam question] **Points**
   Read the whole question before you attempt the parts.

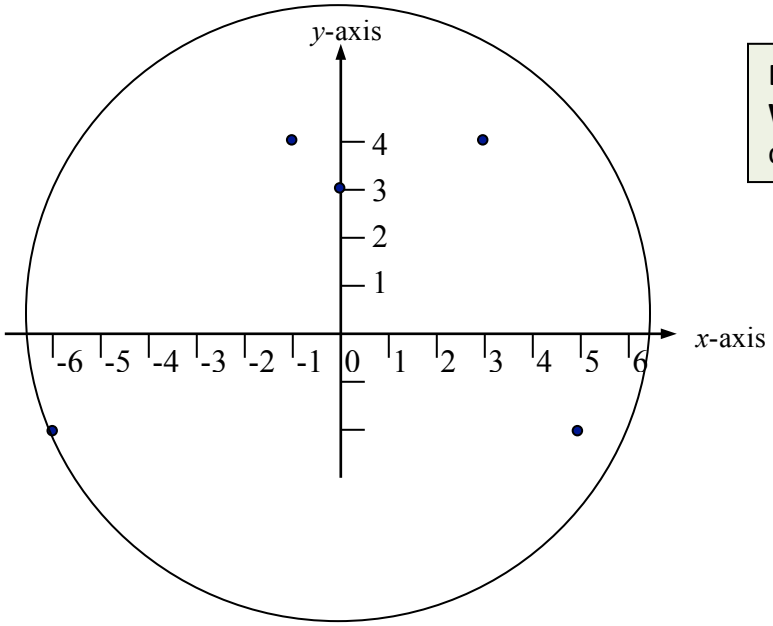
   a. Declare a structure type **point_t** whose members are the *x*-coordinate and *y*-coordinate of a point. The coordinates are integers.

   b. Write a function **read_points()** to read the number of points and points' data into an array of points. Each point is represented by its *x*-coordinate and *y*-coordinate. An example of the input data is shown on the right.

      You may assume that the input data contain at least 1 point and at most 10 points.

```
5
3 4
-1 4
5 -2
-6 -2
0 3
```

   c. Write a function **float circle_area()** to return the area (a floating-point value) of the smallest circle, with centre at the origin (0,0), that encloses all the given points. Assume that π is 3.14159. For our example above, the area is **125.66**.



Download skeleton **Week13_Q4.c** from cs1010 account

   d. Complete the program **Week13_Q4.c** by writing the above mandatory functions. The main function should call the **read_points()** and **circle_area()** functions, and display the area of the circle. You may create additional function(s) of your own.

Answer: See **Week13_Q4.c**

5. **Cumulative Average Point (CAP)**

Write a program **Week13_Q5.c** that makes use of these structures:

- **result_t** that contains 3 members: a 7-character module code, the grade obtained by the student, and the number of modular credits (MCs) of that module.
- **student_t** that contains the student's name (at most 30 characters), and an array of result_t structures. You may assume that a student can take at most 50 modules.

Your program should read in a student's name, the number of modules he has taken, and for each module, the module code, the grade obtained, and the number of modular credits. All these data should be stored in a **student_t** variable. Your program should then compute the student's CAP, based on this formula:

CAP = Σ (MCs × Grade Point) / Σ (MCs)

The table below shows the grade point corresponding to a grade:

Grade	A+ or A	A-	B+	B	B-	C+	C	D+	D	F
Grade Point	5.0	4.5	4.0	3.5	3.0	2.5	2.0	1.5	1.0	0

For example, if Brusco Beh's has taken 5 modules and his results (module code, grade obtained, and number of MCs) are as follows:

CS1010 A+ 4
CS1231 B 4
MA1101R B+ 4
GEM1211 A- 3
PH2001 C 4

Download skeleton **Week13_Q5.c** from cs1010 account

then his CAP is calculated as follows:

(5.0×4 + 3.5×4 + 4.0×4 + 4.5×3 + 2.0×4) / (4 + 4 + 4 + 3 + 4) = 71.5/19 = **3.76**

A sample run is shown below.

```
Enter student's name: Brusco Beh
Enter number of modules taken: 5
Enter results of 5 modules:
CS1010 A+ 4
CS1231 B 4
MA1101R B+ 4
GEM1211 A- 3
PH2001 C 4
CAP = 3.76
```

Answer: See **Week13_Q5.c**

II. File Processing

6. Rewrite your sorting program to read in a list of integers from an input text file, sort the list, and write the sorted list to an output text file. You may make your own assumption on the largest size of your list.

   Suppose the input file is **q6.in** and a sample is shown below. The integer in the first line indicates the size of the list.

```
5
100
-3
8
207
65
```

   A sample run is shown below:

```
Enter input filename : q6.in
Enter output filename: q6.out
```

   Then the output file **q6.out** created will contain the following data:

```
5
-3
8
65
100
207
```

   Download skeleton **Week13_Q5.c** from cs1010 account

   Answer: See **Week13_Q6.c**

7. What if in Q6 above, the file **q6.in** contains user data only (i.e., instead of indicating the size of the list, the integer in the first line is just a data). Modify your **Week13_Q6.c** into **Week13_Q7.c** to correctly read all data and sort them. The output file **q7.out** will contain the following data:

```
5
-3
8
65
100
207
```

   Answer: See **Week13_Q7.c**

III. Revision

8. [10 marks] Study the following C function:

```
int foo(int n) {
    int i, a[3] = {2, 3, 5};

    if (n == 1)
        return 1;

    for (i=0; i<3; i++)
        if ( !(n%a[i]) )
            return foo(n/a[i]);

    return 0;
}
```

Note: using loops inside recursion is out of scope; but this exercise is simple, so suitable for your practice.

(a) [3 marks] Give the sequence of function calls in the order of execution as well as the final return value for the following references:

```
foo(30) ->


foo(840) ->
```

(b) [3 marks] Briefly describe the functionality of `foo()`. Marks will be deducted for long-winded answers.

(c) [4 marks] Give an iterative version of `foo()`. You must complete the function given below and make use of the 'for' loop.

```
int foo_iter(int n) {
    int i, a[3] = {2, 3, 5};

    for (i=0; i<3; i++) {
        /* Fill in your code here. */




    }
    /* Fill in your code here. */


}
```

9. [10 marks] The content of a two-dimensional array `number[6][6]` is given below:

15	22	8	11	59	44
50	64	29	10	34	20
17	86	27	98	57	21
6	16	88	42	36	45
31	26	12	23	82	54
28	66	58	69	41	1

Consider the following code fragment, where selectionSort is the Selection Sort technique covered in lecture.

```
int i, j, tempNumber[6];

for (i = 0; i < 6; i++)
    selectionSort( number[i], 6 );
// At this point, fill in Table (a) on the content in the number array.

for (j = 0; j < 6; j++) {
    for (i = 0; i < 6; i++)
       tempNumber[i] = number[i][j];

    selectionSort( tempNumber, 6 );

    for (i = 0; i < 6; i++)
       number[i][j] = tempNumber[i];
}
// At this point, fill in Table (b) on the content in the number array.
```

Table (a)

Table (b)

Answer: See PowerPoint file

Due to a special property of this sorted number array in Table (b), we can design an algorithm similar to binary search algorithm to search for a key in the array. Below is the recursive algorithm with some parts (4 blank lines) left for you to fill. The function is to be invoked by calling:

```
search( wantedNumber, table, 0, 0, 5, 5 )
```

which returns 1 if `wantedNumber` is in the number array, or 0 otherwise.

```
// search 2D array using binary idea and recursion
int search (int key, int table[][6], int startX, int startY,
            int endX, int endY)
{
    int midX = (startX+endX) / 2,
        midY = (startY+endY) / 2;
    if (startX > endX || startY > endY) return _________ ;
    if (key == table[midX][midY])       return _________ ;
    if (key < table[midX][midY])
        return

              ____________________________________________________
     else
        return

              ____________________________________________________

 }
```

Answer: See PowerPoint file