Tutorial - Week 6- Solution

Objectives: To practise with
- Struct , Union, and Enumeration
- User Defined Data Types

1. **What is the primary difference between a structure and an array? Which would you use to store the catalog description of a course? To store the names of students in the course?**

Solution:

A **structure can** have components/members of **different types**, but an **array's elements must all be of the same type**. Use a structure for the catalog item and an array of strings for the list of student names.

2. **Define a struct data type `date_t` and a function `displayDate`, that will output its `date_t` type parameter in the form `dd/mm/year`.**

Example: Assuming that a variable `currentDate` is of type `date` and its contents is

day	month	year
9	5	2016

The function call `displaydate (currentDate)` will output `9/5/2016`

Solution:

```
/*--data type definition--*/
typedef struct
{
    char day;
    char month;
    int year;
} date_t;

/*-- function prototype --*/
void displayDate(date_t cDate)

/* --function definition--*/
void displayDate( date_t cDate )
{
  printf(" %d/%d/%d", cDate.day, cDate.month,  cDate.year);

  return;
}
```

3. **Considering the following C program segment**

```
typedef struct
{
      char   name[20];
      int    id;
      float  mark[5];

}person_t;

person_t groupOne[10];
```

Indicate whether the following statements are valid or invalid

a. `person_t.id =2907;`
b. `groupOne[5].id = 2645;`
c. `groupOne[0].mark[4]= 45.7;`
d. `printf( "%d\n", groupOne.id );`

Solution:

a. `person_t.id =2907;`
   Invalid as `person_t` is a struct data type and not a struct variable
b. `groupOne[5].id = 2645;`
   Correct as the array element `groupOne[5]` is a variable of `person_t` data type whose one of its members is *id*
c. `groupOne[0].mark[4]= 45.7;`
d. `printf( "%d\n", groupOne.id );`
   Invalid as `groupOne` is an array name and not an array element

4. **a) Define a type named `long_lat_t` that would be appropriate for storing longitude or latitude values. The type comprises components named `degrees` (an integer), `minutes` (an integer), and `direction` (one of the characters 'N' , 'S' , 'E' , or 'W').**
b) The following is a type to represent a geographic location and a variable of this hierarchical structure type. We will assume that `STRSIZ` means 20.

```
typedef struct {
      char place[STRSIZ];
      long_lat_t  longitude,
                  latitude;
} location_t;

location_t resort;
```

Figure 1 gives the content of the variable `resort in the memory`

Variable resort, a structure of type location_t

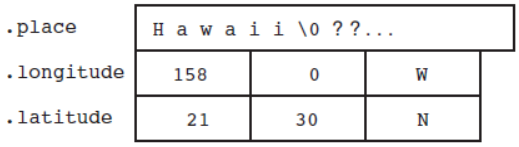

Figure 1

Complete the following table .

Reference	Data Type of Reference	Value
resort.latitude	long_lat_t	21 30 'N'
resort.place	_____	_____
resort.longitude.direction	_____	_____
_____	int	30
resort.place[3]	_____	_____

Solution:

a)

```
[1]typedef struct {
      int degrees;
      int minutes;
      char direction;
} long_lat_t;
```

b)

```
char place[STRSIZ]      "Hawaii"
char        'W'

resort.latitude.minutes

char        'a'     (the 4th char of Hawaii)
```

5. **Write functions `multiply_complex` and `divide_complex` to implement the operations of multiplication and division of complex numbers defined as follows:**

$$(a + bi) \times (c + di) = (ac - bd) + (ad + bc)i$$

$$\frac{(a + bi)}{(c + di)} = \frac{ac + bd}{c^2 + d^2} + \frac{bc - ad}{c^2 + d^2} i$$

Hint: Define a `struct` data type named `complex_t`, with two `double` components, `real` and `imag` to hold the real and imaginary part of the complex number, respectively.

[1] https://gisgeography.com/latitude-longitude-coordinates/

Solution:

```
typedef struct {
double real, imag;
} complex_t;

/*
 *  Returns product of complex values c1 and c2.
 */
complex_t multiply_complex(complex_t c1, complex_t c2)
{
      complex_t cmult;

      cmult.real = c1.real * c2.real - c1.imag * c2.imag;
      cmult.imag = c1.real * c2.imag + c1.imag * c2.real;

      return (cmult);
}

/*
 *  Returns quotient of complex values (c1 / c2).
 */
complex_t divide_complex(complex_t c1, complex_t c2)
{
   complex_t cdiv;
   double denom;

   denom =  c2.real * c2.real + c2.imag * c2.imag;
   cdiv.real = (c1.real * c2.real + c1.imag * c2.imag)/denom;
   cdiv.imag = (c1.imag * c2.real - c1.real * c2.imag)/denom;

   return (cdiv);
}
```

6. **Given the following definitions:**

```
typedef struct {
                char fst_name[20],last_name[20];
                int score;
                char grade;
                } student_t;
. . .
student_t stu1, stu2;
```

   a. **Identify the following statements as possibly valid or definitely invalid. If invalid, explain why.**

      i. `student_t stulist[30];`
      ii. `printf("%s", stu1);`
      iii. `printf("%d %c", stu1.score, stu1.grade);`
      iv. `stu2 = stu1;`
      v. `if (stu2.score == stu1.score)`

```
        printf("Equal");
vi. if (stu2 == stu1)
        printf("Equal structures");
vii. scan_student(&stu1);
viii. stu2.last_name = "Martin";
```

Solution:

i. Valid
ii. Invalid: printf does not accept a C-struct variable.
iii. Valid, access to the members of stu1 using dot operator.
iv. Valid, unlike comparison, C-struct variables of the same datatype can be assigned to each other
v. Valid, comparing the members of C-struct variables (of a primitive data type) is permissible
vi. Invalid: Equality operators cannot be used with C-struct variables
vii. Valid (assuming parameter type is student_t *)
viii. Invalid: Cannot copy strings with = except when intialising the variable during the declaration (to copy a literal string after declaration use the built-in function **strcpy**)

b. Identify the type of each of the following references:

i. `stu1`
ii. `stu2.score`
iii. `stu2.fst_name[3]`
iv. `stu1.grade`

Solution:

i. student_t
ii. int
iii. char, as it is fst_name which is an array of char
iv. char

c. Write a statement that displays the initials of `stu1` (with periods).

Solution:

```
printf("%c.%c.", stu1.fst_name[0],stu1.last_name[0]);
```

d. How many components does variable `stu2` have?

Solution:

Four

e. Declare an array of 40 `student_t` structures, and write a code segment that displays on separate lines the names (*last name, first name)* of all the students in the list.

Solution:

```
student_t students[40];
for (i = 0; i < 40; ++i)
   printf("%s, %s\n", students[i].last_name,
              students[i].fst_name);
```

f. Write functions `scan_student` and `print_student` for type `student_t` variables.

Solution:

The following code gives the entire program and not just the requested function defintions.

```
#include<stdio.h>

typedef struct {
                char fst_name[20],last_name[20];
                int score;
                char grade;
                } student_t;

student_t scan_student();
void print_student(student_t stu);

int main(void){
student_t st;
st=scan_student();
if(st.score!=-1)     print_student(st);
else printf("Error\n");
}

student_t scan_student()/* output - student structure to fill */
{
student_t error;
error.score=-1;
student_t stu;

int status;
status = scanf("%s%s%d %c", stu.fst_name,stu.last_name,&stu.score, &stu.grade);
// space before %c is needed to "swallow" the white space after user's input integer
if (status == 4) {
return (stu);
}
else  {
return error;
}
}


void print_student(student_t stu) /* input - student structure
                                  to display */
{
printf("Student: %s, %s\n", stu.fst_name,stu.last_name);
```

```
printf(" Score: %d Grade: %c\n", stu.score,stu.grade);
}
```

7. **What output is produced by the following program?**

```
typedef union    /* can store only one value at a time  */
{
  char   var1;
  int    var2;
  double var3;
} mixData_t;


int main(void)
{
   mixData_t myData;

   myData.var1 = 'A';
   myData.var2 = 35;
   if(myData.var1 == 'A')
     myData.var3 = 5.0;
   else myData.var3 = -1.5;

    printf ("%f", myData.var3);
}
```

Solution:

Output: -1.5 as the character 'A' gets overwritten by 35 when setting var2 member value; therefore, the else statement gets executed

8. **Electromagnetic spectrum is subdivided into several bands:**

LF, MF, HF, VHF, UHF

Define a new data type `band_t` that can take only these values. Declare array `channels` of type `band_t` that can store 44 elements. Write a function `initChannels()` that initializes all elements of the array with the value VHF using a for loop.

```
/*--data type definition--*/

typedef enum { LF, MF, HF, VHF, UHF } band_t;
```

```
#define NUM_OF_CHAN          44

/* -- array of bands --*/
band_t channels [ NUM_OF_CHAN];

/*-- function prototype --*/
void initChannels(band_t chan[], int size );

/* --function definition--*/
void initChannels(band_t chan[], int size )
{
    int i;
    for(i=0; i<size; i++)
    chan[i] = VHF;
    return;
}
```

9. **Write a function `print_day` for enumerated type `day_t` that displays its argument as a string.**

`void print_day(day_t cur_day);`

Hint: **Use a `switch` statement to select the appropriate `printf` statement.**

```
typedef enum {SUN, MON, TUE, WED, THU,FRI,SAT} day_t;

    void print_day(day_t cur_day)
    {
      switch (cur_day) {
        case SUN: printf("Sunday\n");
                  break;

        case MON: printf("Monday\n");
```

```
            break;

        case TUE: printf("Tuesday\n");
            break;

        case WED: printf("Wednesday\n");
            break;

        case THU: printf("Thursday\n");
            break;

        case FRI: printf("Friday\n");
            break;

        case SAT: printf("Saturday\n");
      }
   }
```

To test it, you can write

```
int main(void){
   day_t cur=SUN;
   print_day(cur);
}
```