IT1007 Introduction to Programming with Python and C
C Lab Exercise 04

Submission instructions [Coursemology]:

1. There are two parts in this lab. For Part A, you have to submit within the same day of your lab session. For your Part B, you have six days to work on it. (E.g. if your TLab is on Monday, then your deadline is the coming Sunday midnight.)
2. Complete your code using the skeleton files provided, then **test your code on your computer first** before submitting to Coursemology.
3. To submit your code on Coursemology, click on "Labs" in the sidebar followed by the appropriate "Attempt" button.
4. **Copy ONLY the required function** from your completed skeleton file into the Coursemology code window.
5. Click "Run Code" to test that your function works on Coursemology.
6. Click "Finalise Submission" to submit your code for the **ENTIRE Part A/B**. **You will not be able to amend your code after you have finalised your submission.**
7. You must name your functions exactly as the questions state.

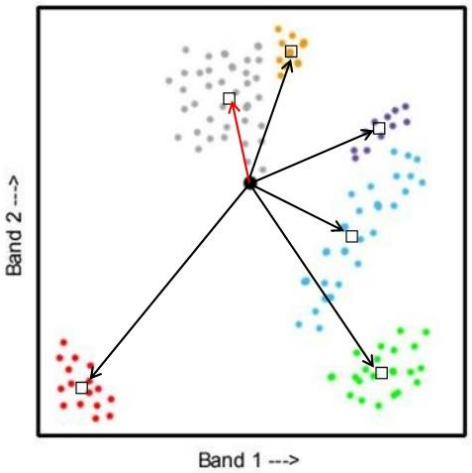
Failure to follow each of the instruction will result in 10% deduction of your marks.

Part A (Deadline: Same day of your TLab)

Back during the days of Python lab, we have defined the color green as the RGB values where G > R, G > B and G > 110. This is adequate for a single color. For classification of a wider spectrum of colors, we can use a simple concept called the *minimum Euclidean distance classifier*, which is used in machine learning to categorize new data using pre-existing labelled data.

In a *minimum distance classifier*, points that are closer together are of the same category; a random color point will most likely share a similar color to the point that lies closest to it. We can plot these color points on a three-dimensional vector space with red, green and blue axes, and the magnitude of the distance between any two points is then given by the Euclidean distance between them:

$$Distance = \sqrt{(R_1 - R_2)^2 + (G_1 - G_2)^2 + (B_1 - B_2)^2}$$



By finding the distance of a given RGB point to every available color data provided, the color point whose distance is the minimum out of all of the data is the classified color of the given RGB point. In the picture on the right, the distance of the black dot is closest to the grey dots and is hence classified as "grey".

Hint: To find out the color of a given point, just do a quick Google search, e.g. "RGB 206 10 40".
Source: https://de.wikiversity.org/wiki/Datei:Minimum_Distance.jpg

Question 1: Euclidean Distance (10 Marks)

The function definition for **find_distance** in Python below takes in two RGB points (represented as a list [R, G, B]), **pixel** and **sample**, and returns the distance between them as a float.

```
def find_distance(pixel, sample):
    # pixel / sample = [R, G, B]
    red_sq = (pixel[0] - sample[0])**2
    green_sq = (pixel[1] - sample[1])**2
    blue_sq = (pixel[2] - sample[2])**2
    return (red_sq + green_sq + blue_sq)**0.5
```

The structure of an RGB point in C has been defined for you as follows:

```
struct RGB {
    int red;
    int green;
    int blue;
};
```

TASK

Convert the following function definition for **find_distance** from Python into C, replacing the list type of pixel and sample in Python into a struct RGB type in C.

```
#include <stdio.h>
#include <math.h> // for pow (returns double)

float find_distance(struct RGB pixel, struct RGB sample) {
    // Write your code here
}
```

TEST CASES

```
struct RGB yellow = {255, 255, 0};
struct RGB blue = {0, 0, 255};
struct RGB azure = {0, 128, 255};
```

Input	Expected output
find_distance(azure, yellow);	382.333618
find_distance(azure, blue);	128.000000
find_distance(azure, azure);	0.000000

Question 2: Printing Points (10 Marks)

A well-defined color has a name associated with its RGB values. The structure for color is simply an extension of the RGB structure, and has been defined for you as follows:

```
struct color {
    char name[20];
    struct RGB value;
};
```

TASK

Define the function **print_color** that takes in a struct color, and prints its RGB value and name in the following format:

```
RGB value (255, 0, 0) is the color red.
```

TEST CASES

```
struct color C1 = {"yellow", {255, 255, 0}};
struct color C2 = {"blue", {0, 0, 255}};
struct color C3 = {"azure", {0, 128, 255}};
```

Input	**Expected output**
`print_color(C1);`	`RGB value (255, 255, 0) is the color yellow.`
`print_color(C2);`	`RGB value (0, 0, 255) is the color blue.`
`print_color(C3);`	`RGB value (0, 128, 255) is the color azure.`

Question 3: Minimum Distance Classifier (15 Marks)

We can now start classifying any color based on its distance to pre-existing defined colors. A global array of struct color **color_data** has been defined, which will be used to classify our colors.

```
struct color color_data[27] = {
    {"black", {0, 0, 0}},
    {"navy", {0, 0, 128}},
    {"blue", {0, 0, 255}},
    {"green", {0, 128, 0}},
    ...
    {"white", {255, 255, 255}}
};
```

TASK

Define the function **identify_color** that takes in a struct RGB, and prints its RGB values as well as the name of the color point in the array **color_data** that is closest to it, in the following format (same as that of **print_color**!):

```
RGB value (0, 0, 254) is the color blue.
```

The functions **find_distance** and **print_color**, as well as struct RGB and color have already been defined for you. You may assume that the RGB input will not map to more than one color from **color_data**. Hint: To obtain the minimum value, iterate through the array of colors and compare the calculated distance from the previous iteration.

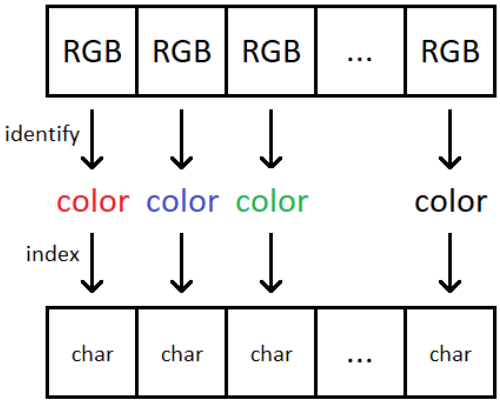
TEST CASES

```
struct RGB c1 = {0, 0, 254};
struct RGB c2 = {79, 100, 8};
struct RGB c3 = {200, 120, 40};
```

Input	**Expected output**
`identify_color(c1);`	`RGB value (0, 0, 254) is the color blue.`
`identify_color(c2);`	`RGB value (79, 100, 8) is the color olive.`
`identify_color(c3);`	`RGB value (200, 120, 40) is the color orange.`

Question 4: Extra – Image Compression (0 Marks)

Uncompressed images occupy a lot of storage space because every pixel in the array of colors is defined with an individual set of RGB values – a struct RGB occupies a staggering 12 bytes since each int takes up 4 bytes!



A naïve method of compressing such an image involves converting each pixel in the uncompressed image into a char index of a pre-defined color dataset. The compressed image then stores these indices in a char array with the same dimensions. This achieves a compression ratio of approximately 9%, since char is only 1 byte each.

Consider an uncompressed one-dimension array of RGB values called **uncompressed_pic**, from which we will compress and store in a (compressed) one-dimensional array of char called **compressed_pic**. Each pixel from **uncompressed_pic** is identified as a color from **color_data**, and is translated to a char equal to the index in which the color appears in the array **color_data**, which is then stored in the corresponding position in **compressed_pic**.

TASK

Define a function **compress_image** that takes in a struct RGB array (compressed image), a char array (uncompressed image) and an int corresponding to the size of the arrays, and performs the compression algorithm as described above.

You may assume that the sizes of both arrays are equal. The function **find_distance** as well as struct RGB, struct color and **color_data** has been defined for you.

TEST CASES

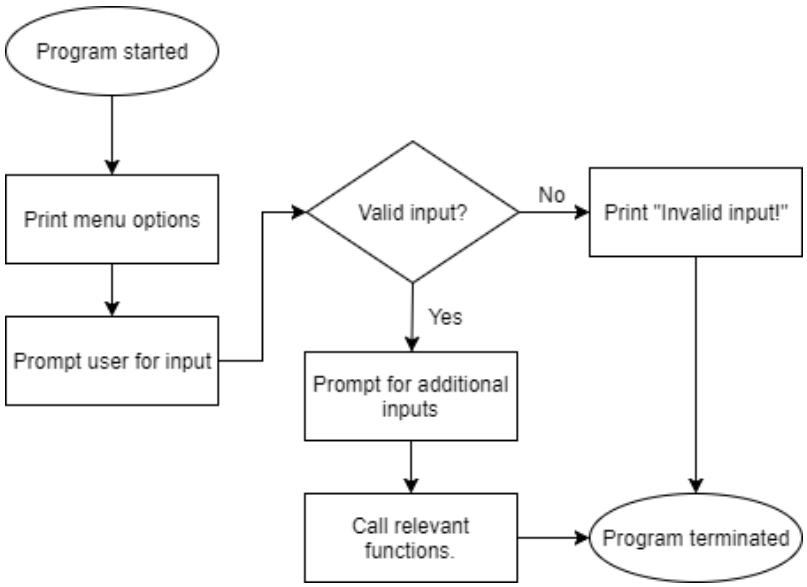
```
char nice_numbers[6];

struct RGB nice_colors[6] = {
    {0, 2, 3},
    {9, 240, 18},
    {200, 2, 3},
    {9, 24, 254},
    {200, 140, 70},
    {130, 109, 244}
};
```

Input	Expected output
`compress_image(nice_colors, nice_numbers, 6);`	`[0, 6, 18, 2, 22, 14]`

Part B (Deadline: Day of TLab + 6 days)

In electrical engineering, different electronic components must be selected such that the cost, performance and compatibility with other hardware is balanced. In this part, we will guide you to implement an electronic device database (EDD) that allows us to populate it with different device data, and write a program that can search for suitable devices to facilitate the selection of such components.

In particular, we want to be able to find out the size of the database (**database_info**), search for devices by its port type or electrical ratings (**search_port_type**, **search_current_power**), and print the search results (**print_device**). The flowchart of the program is illustrated below.



General instructions: Read the entire problem carefully before you begin designing your solution. You may consider declaring all the arrays as global arrays. Do not assume the size of the database.

Question 1: Populating and printing the database (11 marks)

Each device structure is defined to have the following format as specified below:

```
struct device {
    int ports; // Number of ports
    char port_type; // Type: A / B / C / D
    float cost; // Device cost
    int cpu_cores; // Number of CPU cores
    int chips; // Numbers of chips in this device
    struct {
        int version;
        int year;
        char manufacturer[10]; // e.g. Micron, Intel
    } make;
    float current_rating, power_rating;
};
```

TASK

Create a global device array **dev_database** for storing information on the following 5 devices (the brackets in between capture data for the sub-structure). Manually enter this data given.

Device 1: 5, B, 73.45, 4, 7, {2, 2015, Intel}, 3.1, 1.2
Device 2: 5, A, 70.00, 2, 5, {1, 2010, Micron}, 2.0, 1.5
Device 3: 5, B, 65.00, 2, 5, {1, 2011, Intel}, 3.0, 1.5
Device 4: 5, C, 80.00, 8, 7, {1, 2015, Motorola}, 2.5, 1.0
Device 5: 5, A, 70.00, 2, 5, {3, 2012, Intel}, 3.0, 1.3

Also define a function **print_device** that takes in an int and prints the device information of the struct device located at that index of **dev_database**, in the following format:

```
-----
Intel (2015) Version 2

Ports = 5 x B
Cost = $73.45
CPU cores = 4
Chips = 7
Rating = 3.1A / 1.2W
```

An additional 1 mark will be awarded for appropriate comments.

TEST CASES

Input	Expected output

`print_device(0);`	`-----`
	`Intel (2015) Version 2`
	
	`Ports = 5 x B`
	`Cost = $73.45`
	`CPU cores = 4`
	`Chips = 7`
	`Rating = 3.1A / 1.2W`

`print_device(4);`	`-----`
	`Intel (2012) Version 3`
	
	`Ports = 5 x A`
	`Cost = $70.00`
	`CPU cores = 2`
	`Chips = 5`
	`Rating = 3.0A / 1.3W`

Question 2: Measuring database size (6 marks)

TASK

Define a function **database_info()** that takes in no arguments and prints the storage size (number of bytes) required to store each device, as well as the number of elements that the device array **dev_database** can store:

```
Storage size for each device = 48 bytes
Number of devices that can be stored = 5
```

An additional 1 mark will be awarded for appropriate comments. **dev_database** has already been defined for you. Since we may want to increase the size of **dev_database** when deploying the program with more device information, do not assume that the size is necessarily 5.

TEST CASES

Input	Expected output
`database_info();`	`Storage size for each device = 48 bytes` `Number of devices that can be stored = 5`

Question 3: Search by port types (16 marks)

TASK

Define a function **search_port_type** that takes in a char representing the device port type (i.e. 'A', 'B', 'C' or 'D') and prints the device information for all devices (if any) that share the same port type using **print_device** (starting from the first device in **dev_database**), as well as the number of devices found prior to printing the device information.

An additional 1 mark will be awarded for appropriate comments. Both **dev_database** and **print_device** have already been defined for you.

TEST CASES

Input	**Expected output**

`search_port_type('B');`	`2 device(s) found!`
	`-----`
	`Intel (2015) Version 2`
	
	`Ports = 5 x B`
	`Cost = $73.45`
	`CPU cores = 4`
	`Chips = 7`
	`Rating = 3.0A / 1.3W`
	`-----`
	`Intel (2011) Version 1`
	
	`Ports = 5 x B`
	`Cost = $65.00`
	`CPU cores = 2`
	`Chips = 5`
	`Rating = 3.0A / 1.5W`

`search_port_type('D');`	`0 device(s) found!`

Question 4: Search by current / power ratings (16 marks)

TASK

Define a function **search_current_power** that takes in two floats (current and power respectively) and prints the device information for all devices (if any) that share the same current and power rating, as well as the number of devices found prior to printing the device information.

An additional 1 mark will be awarded for appropriate comments. Both **dev_database** and **print_device** have already been defined for you.

TEST CASES

Input	**Expected output**

`search_current_power(2.5, 1.0);`	`1 device(s) found!`
	`-----`
	`Motorola (2015) Version 1`
	
	`Ports = 5 x C`
	`Cost = $80.00`
	`CPU cores = 8`
	`Chips = 7`
	`Rating = 2.5A / 1.0W`

Question 5: EDD Program (16 marks)

TASK

Complete the main function so that the user can search for devices in **dev_database**.

1. When the program is first started, a list of menu options is provided. The user is then prompted for input using the following prompt below (with a space after the colon : mark).

```
IT1007 EDD
1 = Get database information
2 = Search by port type
3 = Search by current / power ratings
=====
Input:
```

2. If the user input is valid, prompt the user to input the required additional information if required, according to the specification below (with a space after colon : marks). After the additional inputs are obtained, simply call the required functions with the appropriate arguments. If user input is invalid, print "Invalid input!".

```
// For input 1
// Simply print the database information

// For input 2
Specify port type:
// Print relevant device(s) information

// For input 3
Specify current and power:
// Print relevant device(s) information

// For invalid inputs
Invalid input!
```

An additional 1 mark will be awarded for appropriate comments. All the necessary functions and arrays defined previously are already defined for you in this question, i.e. **dev_database**, **print_device**, **database_info**, **search_port_type**, **search_current_power**.

TEST CASES

Input	Expected output
Sample run of main program with invalid input	IT1007 EDD 1 = Get database information 2 = Search by port type 3 = Search by current / power ratings ===== Input: D Invalid input!
Sample run of main program with inputs 2 and B	IT1007 EDD 1 = Get database information 2 = Search by port type 3 = Search by current / power ratings ===== Input: 2 Specify port type: B 2 device(s) found! ----- Intel (2015) Version 2 Ports = 5 x B Cost = $73.45 CPU cores = 4 Chips = 7 Rating = 3.0A / 1.3W ----- Intel (2011) Version 1 Ports = 5 x B Cost = $65.00 CPU cores = 2 Chips = 5 Rating = 3.0A / 1.5W