The University of Melbourne
School of Computing and Information Systems
COMP20005 Engineering Computation
Semester 2, 2018
Assignment 2
Due: 4:00pm Wednesday 17th October 2018

1 Learning Outcomes

In this assignment you will demonstrate your understanding of arrays and structures, and use them together with functions. You will further practise your skills in program design and debugging.

2 The Story...

Wi-Fi access has become a must-have at any modern venue such as shopping malls, airports, university campuses, etc. Wi-Fi signal strength plays a vital role in the experience of staying at these venues.

Your task in this assignment is to design and implement a program that models the Wi-Fi signal strength pattern across a given region, and calculate any zones that perceive an insufficient level of signal strength.

You do *not* need to be an expert in wireless communication, but you will need the following background information to carry out this assignment. When measuring the signal strength at a certain location, we need to know the *transmission power* of the *Wi-Fi access points* (WAP, e.g., a Wi-Fi router). Wi-Fi signal transmission power, like other electromagnetic waves, is usually measured in a unit called *decibel-milliwatts (dBm)*. For example, the typical wireless transmission power of home Wi-Fi routers is 20 dBm. For enterprise Wi-Fi routers, the transmission power can be as high as 30 dBm. The distance to the WAP also plays an important role because the signal strength reduces quickly with the increase of distance to the WAP[1]. There are a number of models to formulate the signal strength reduction with regard to the distance to the WAP. In this assignment we use a simple one, called the *free-space path loss* (FSPL) model. This model calculates the loss in signal strength through free space (usually air), with no obstacles nearby to cause reflection or diffraction[2]. Given a WAP with a signal frequency of f GHz, the signal reduction at a distance of d metres is calculated by the FSPL model as follows[3]:

$$FSPL(d, f) = 20\log_{10}(d) + 20\log_{10}(f) + 32.45. \tag{1}$$

For example, given a WAP with a signal frequency of $f = 5$ GHz, the loss of signal strength for a distance of $d = 10$ metres is calculated as:

$$FSPL(10, 5) = 20\log_{10}(10) + 20\log_{10}(5) + 32.45 \approx 20 \times 1 + 20 \times 0.70 + 32.45 = 66.45.$$

Assume that the WAP has a transmission power of 30 dBm. Then the signal strength at 10 metres from the WAP is $30 - 66.45 = -36.45$ dBm. Similarly, the loss of signal strength for a distance of $d = 100$ metres is 86.45. The signal strength at 100 metres from the WAP is $30 - 86.45 = -56.45$ dBm.

[1] The background noise is another important factor, but we ignore it in this assignment to simplify the computation model.
[2] `https://en.wikipedia.org/wiki/Free-space_path_loss`
[3] Strictly speaking, this equation works with dB instead of dBm. Nevertheless, we use it here for simplicity.

3 Your Task

3.1 Stage 1 - Arrays and Structs (marks up to 5/20)

Write a program that reads a sequence of transmission power, signal frequency, and coordinates (x and y) from the standard input into an array of a `struct` type. Each relevant input line will start with a '`P`' (for WAP) in the first character position. For example:

```
P 20.0 5.0 10.1 0.1
P 29.0 4.1 10.1 50.1
P 23.0 3.2 35.1 75.1
```

Here, we have three WAPs. Column 1 is the label '`P`' indicating that this line contains information of a WAP. Columns 2 and 3 are the transmission power (in dBm) and signal frequency (in GHz) of the WAPs. Columns 4 and 5 are the x- and y-coordinates in the positive quadrant of the plane measured in metres from the origin $(0, 0)$.

All input lines starting with a '`P`' *will appear together at the beginning of the data file.*

Once the WAP data have been read, your program should calculate the *maximum signal strength* at the origin (location $(0, 0)$) as follows. Assume that there are n WAPs $P_1, P_2, ..., P_n$. Their signal strength at the origin (calculated based on Equation 1) are $s_1, s_2, ..., s_n$, respectively. The maximum signal strength at the origin is the maximum among $s_1, s_2, ..., s_n$, that is, $\max\{s_1, s_2, ..., s_n\}$.

For the three WAPs (that is, $n = 3$) in the sample input, your output for this stage should be:

```
Stage 1
==========
Number of WAPs: 03
Maximum signal strength at (00.0, 00.0): -46.52 dBm
```

Note that Equation 1 requires $d \neq 0$. To guarantee the validity of the Equation, *the WAP points will not locate at the origin or any of the points used in the following tasks.* You may assume between 1 and 99 (inclusive) WAPs in the input data. All input data will be valid and correctly formated.

3.2 Stage 2 - More Loops (marks up to 10/20)

The second block of input data are lines starting with an '`O`', where each line contains the x- and y-coordinates of an observation point. Your program should read each of these lines, and compute the maximum signal strength at each observation point, based on the WAPs that were read in Stage 1. For example, if the next two input lines are:

```
O 13.0 29.0
O 38.0 41.0
```

your Stage 2 output, following on from your Stage 1 output, should be:

```
Stage 2
==========
Maximum signal strength at (13.0, 29.0): -42.27 dBm
Maximum signal strength at (38.0, 41.0): -45.06 dBm
```

You may assume between 1 and 99 (inclusive) observation points in the input data. *All input lines starting with an* '`O`' *will appear together after the lines starting with a* '`P`'.

3.3 Stage 3 - Yet More Loops (marks up to 15/20)

To estimate the overall Wi-Fi signal strength of a given region, the simplest approach is to apply a regular grid, and evaluate the signal strength at each point in the grid. For this stage, assume that the region is a square of 78×78 metres, with edges perfectly aligned north-south and east-west. Add further functions and loops to your program so that it evaluates the maximum perceived signal strength at every 1-metre grid point strictly within the region (that is, at the points $x \in \{1, 2, ..., 77\} \times y \in \{1, 2, ..., 77\}$), and counts the percentage of those points at which the maximum perceived signal strength is lower than or equal to a -45 dBm threshold which is considered to be too low.

With the same three sample input WAPs shown in Stage 1, the next section of the output should be:

```
Stage 3
==========
5929 points sampled
3588 points (60.52%) with maximum signal strength <= -45 dBm
```

Note that the output number of points (e.g., `3588`) with maximum signal strength lower than or equal to the -45 dBm threshold should be printed with formatter "%04d".

3.4 Stage 4 - Draw a Map (marks up to 20/20)

In this stage, you need to draw a "signal strength map" with a grid of characters. Each character displayed represents a cell of 1 metre wide (in the east-west direction) and 2 metres tall (in the north-south direction), with the 1:2 ratio (roughly) corresponding to the relative width and height of characters in a terminal font. To represent a square region of 78×78 metres, your map will be 78 characters wide and 39 characters high. To show the signal strength contours, you use the following relationship between maximum perceived signal strength in dBm and the character displayed (where ' ' represents a whitespace character):

```
>0    >-5   >-15   >-25   >-35   >-45   <=-45
'+'   ' '    '1'    ' '    '3'    ' '    '-'
```

The character plotted in each cell should correspond to the maximum signal strength at the centre of that cell. For example, the first value to be output (in the top left corner) should correspond to the maximum signal strength at the point $(x, y) = (0.5, 77.0)$, since each cell is taken to be 1 metre wide and 2 metres high. The bottom left corner of your map corresponds to a cell whose midpoint is at $(x, y) = (0.5, 1.0)$. Similarly, the top right and the bottom right points to be plotted are $(77.5, 77.0)$ and $(77.5, 1.0)$, respectively. A sample of the expected output is linked from LMS.

3.5 Stage 5 - Polygonal Boundaries (marks up to 20/20)

This stage is for a challenge. You do *not* need to complete this stage to obtain the full mark of the assignment. Before starting this stage, we suggest to submit (and preserve in a different directory) a program covering Stages 1 to 4 first.

There is 1 bonus mark for this stage. The bonus mark earned in this stage will compensate for the marks you lost in the earlier stages (assuming that you lost some, your total mark will not exceed 20).

Read a third segment of data, starting with an '`X`' at each line, which is followed by the x- and y-coordinates of a vertex on the boundary of the bounding polygon of a given region (Note: the last vertex should be connected back to the first vertex). For example, the input

```
X 0.0 0.0
X 10.0 45.0
X 50.0 75.0
X 75.0 25.0
```

describes a boundary that is an irregular quadrilateral. Your task is to print a second map that only shows the cells for which the cell centroid lies within the polygonal boundary; all other cells should be plotted as '`#`'. You may assume between 3 and 99 (inclusive) vertices in the input data. *All input lines starting with an* '`X`' *will appear together after the lines starting with an* '`O`'.

Hint: you may assume that the polygon is convex, and hence that the centroid of the boundary points lies inside the polygon. A cell should be plotted only if the line segment between its centroid and the *centroid of the polygon* does not intersect any of the boundary line segments (While technically not the case for all polygons, an approximation which takes the "average" of the polygon vertices is good enough to calculate the centroid of the polygon for the assignment). Hence, a critical component of your program needs to be a function that takes two line segments, described by two pairs of points, and determines if they touch in any way. That function will be supplied on the LMS page for the project, and you may copy and paste it into your program (and make sure to acknowledge the use of external code). Samples of the expected output will be given on the LMS page. There will of necessity be some overlap in control structure between this function and your Stage 4 function; that duplication will be tolerated.

4 Submission and Assessment

This assignment is worth 20% of the final mark. A detailed marking scheme will be provided on the LMS.

You need to submit all your code in one file named `assmt2.c` for assessment. The submission process is similar to that of Assignment 1. You will need to log in to a Unix server (dimefox.eng.unimelb.edu.au or nutmeg.eng.unimelb.edu.au) and submit your files using the following command:

```
submit comp20005 a2 assmt2.c
```

You can (and should) use submit both **early and often** - to get used to the way it works, and also to check that your program compiles correctly on our test system, which has some different characteristics to the lab machines. You should verify your submission using the following commands:

```
verify comp20005 a2 > receipt-a2.txt
more receipt-a2.txt
```

Note that the verification output format of this assignment is slightly different from that of Assignment 1. In this assignment, the "`Diff output`" only shows the lines in which your submission output is different from our expected output. The lines starting with a '`<`' is our expected output, while the lines starting with a '`>`' is your submission output. Your should observe such lines closely, find out their differences, and fix any error there may be. If the "`Diff output`" is blank, it means that your submission output is exactly the same as our expected output. We use this different format due to the larger output size of this assignment.

You will be given a sample test file `test0.txt` and the sample output `test0-output.txt`. You can test your code on your own machine with the following command and compare the output with `test0-output.txt`:

```
mac: ./assmt2 < test0.txt    /* Here '<' feeds the data from test0.txt into assmt2 */
```

Only the last submission made before the deadline will be marked. You may discuss your work with others, but what gets typed into your program must be individual work, **not** copied from anyone else. Do **not** give hard copy or soft copy of your work to anyone else; do **not** "lend" your memory stick to others; and do **not** ask others to give you their programs "just so that I can take a look and get some ideas, I won't copy, honest". The best way to help your friends in this regard is to say a very firm "no" when they ask for a copy of, or to see, your program, pointing out that your "no", and their acceptance of that decision, is the only thing that will preserve your friendship. *A sophisticated program that undertakes deep structural analysis of C code identifying regions of similarity will be run over all submissions in "compare every pair" mode.* See https://academichonesty.unimelb.edu.au for more information.

Deadline: Programs not submitted by **4:00pm Wednesday 17th October 2018** will lose penalty marks at the rate of three marks per day or part day late. Late submissions after 4:00pm Friday 19th October 2018 will **not** be accepted. Students seeking extensions for medical or other "outside my control" reasons should email the lecturer at jianzhong.qi@unimelb.edu.au. If you attend a GP or other health care professional as a result of illness, be sure to take a Health Professional Report form with you (get it from the Special Consideration section of the Student Portal), you will need this form to be filled out if your illness develops into something that later requires a Special Consideration application to be lodged. You should scan the HPR form and send it in connection with any non-Special Consideration assignment extension requests.

And remember, *C programming is fun!*