UNIVERSITY OF BRITISH COLUMBIA
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
CPEN 513: CAD Algorithms for Integrated Circuits
2019/2020 Term 2

Assignment 1

Due: Monday January 27, by 11:59pm on the Canvas site

In this assignment, you are to implement two versions of a routing algorithm: (a) The first version will implement the Lee-Moore Shortest Path Routing Algorithm, and (b) The second version will implement an A* algorithm. Your program will read in placement and netlist information, and output the routing and indication of whether the routing was successful.

In class, we made the assumption that each wire connects exactly one source to one sink. It is more realistic to assume a wire can connect one source to *one or more* sinks. Your router is to support this more realistic case. There are several ways to extend the algorithm discussed in class to handle multiple sinks. It is up to you to choose how you want to do it.

Your goal is to maximize the number of segments that can be routed for each benchmark circuit. If you want to extend the work to try to get some of the "initiative marks", try to think about intelligent algorithms to order nets or to rip-up and re-route nets. You should not have a hard-coded net order optimized for each benchmark circuit (that would be too easy!).

Input:
Your program should read in a file that indicates:

- The grid size
- List of grid points that are covered by cells (obstructions)
- A list of wires, indicating, for each wire, which grid points are to be connected

An example input file is shown below.

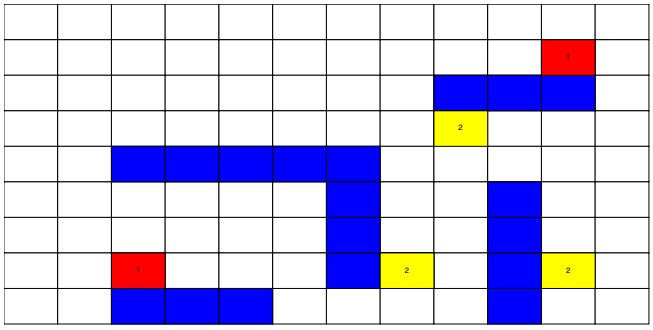
```
12 9
18
8 2
9 2
10 2
2 4
3 4
4 4
5 4
6 4
6 5
6 6
6 7
2 8
3 8
4 8
9 5
9 6
9 7
9 8
2
2 10 1 2 7
3 8 3 7 7 10 7
```

The first line of this file (12 9) indicates the grid size to use (12x9 grid indexed from [0,0] to [11,8]). The second line (18) indicates the number of obstructed cells due to grids. The grid locations that are obstructed are then listed, one location per line. For example, line 3 indicates that grid location (8,2) is occupied by a cell and can not be used for routing. In this example, there are 18 such obstructed cells, ending in cell (9,8).

Following the obstructed cells is the number of wires to route. In this example, there are 2 wires that must be routed. Then, the wires are listed, one per line. Each wire contains at least 5 numbers. The first number indicates the number of pins in this wire (a wire connecting a single source to a single sink would have 2 pins). In the example, the first wire has 2 pins, while the second wire has 3 pins (so it has one source and two sinks). After the number of pins, the source is listed followed by all the sinks. So in the example, the first wire's source is (10,1) and its sink is (2,7). The second wire's source is (8,3) and its two sinks are (7,7) and (10,7).

A diagram showing this graphically is on the next page.

This diagram shows the previous example:



Output:
Your program should have a graphical interface that displays the progress of the router as it is solving the problem. Exactly what to display is up to you, but note that just showing the final solution is not enough; you must show the progress of the algorithm as it proceeds. After all wires have been attempted, your program should indicate how many of the wires were successfully routed.

Benchmarks:
You will test your circuit on the benchmark circuits available on the Canvas web site. For convenience, a graphical representation of each is shown in the appendix of this document. Warning: I may test it on some additional circuits of my own devising, so make sure your code is robust!

What to hand in:
You must handin your files using the Canvas web site. You should hand in a zip file containing the following:

- A text of PDF file (report.pdf or report.txt) containing:
    - Your report should address each of the marking criteria (see attached) clearly. In particular, you should have a description of how your program works (assuming I already have a basic idea of what the Lee Moore and A* algorithms do). Indicate how you handle multiple sinks. Clearly describe how you tested it (just running it on our benchmark circuits and eyeballing the results is not enough for full marks). Indicate any extension beyond the assignment handout that you have implemented.
    - A table showing how many segments can be routed in each of the benchmark circuits on the Canvas web site, for both the Lee-Moore and A* versions of your algorithm.
    - Pictures showing the final routing for stanley.infile and stdcell.infile for both versions of your algorithm (if you are unable to include the pictures in your document, separate files are ok).
    - An indication of how the program can be run. If you are using a departmental Linux/Unix machine, include the full path-name to the executable, and make sure the permissions are set properly (global readable/executable).

- The source code. A link to an on-line repository is best. If you are using a departmental linux/unix machine, you can just tell me the path-name to your source code (just make sure it is globally readable/writable). Alternatively, you can include the source code in your zip file.

Appendix A:EasyGL Graphics Package:

If you are using a Linux station for your assignment, you can optionally use the EasyGL graphics package for graphics. You can download the tar file from the Canvas site. You can unpack the tar file using the following command:

```
tar –xf easygl.tar
```

The graphics package is in graphics.c and graphics.h. An example program using the graphics package is in example.c This program works as follows. It first initializes the graphics environment by calling init_graphics, init_world, and update_message. Then, it calls event_loop with two parameters: one indicating what function to use to draw the screen, and the other to indicate what function to call when the user presses a button. The event_loop function returns when the user hits "Proceed" from the on-screen menu.

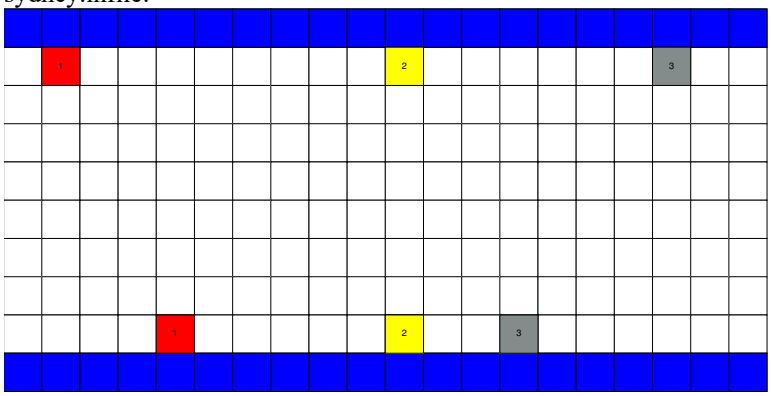
To test the package, type "make". This compiles the graphics package and the example program. If you get errors regarding Xpointer, you may have to remove the comments on line 72 of graphics.c. The executable is placed in the file "example". To get started, you can keep the directory structure as it is, and replace the code in example.c with your code.

Note that if you don't have an ECE account and want one, please contact IT Services (help@ece.ubc.ca) and tell him you are taking this course.

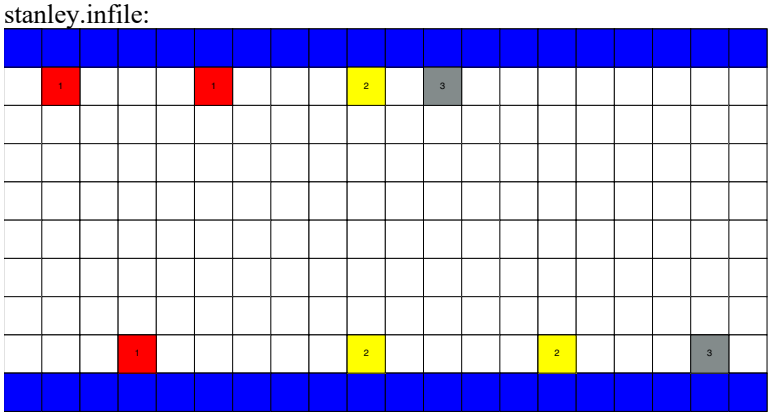
Appendix B: Benchmark Circuits on the Connect website:

The following shows a graphical representation of each of the benchmark circuits. You don't actually need this (since your program will draw graphics) but it might help as you are debugging.

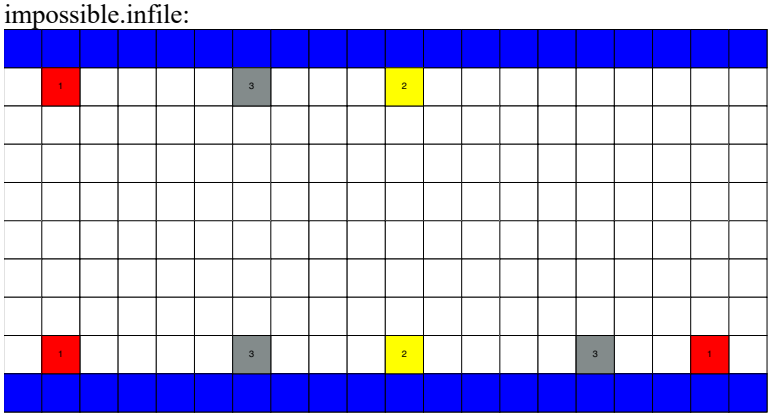
sydney.infile:



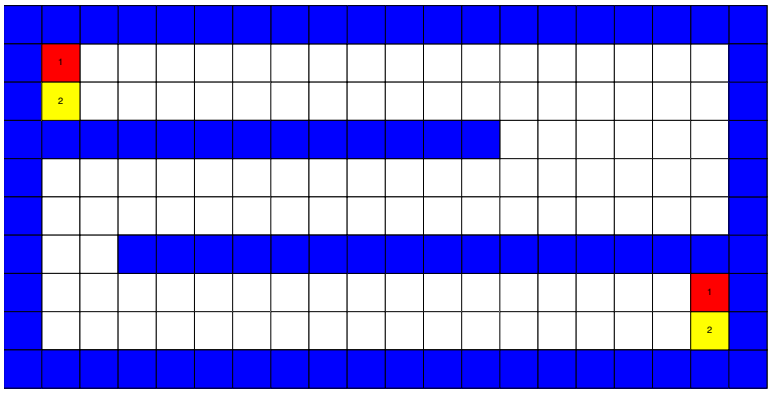
stanley.infile:



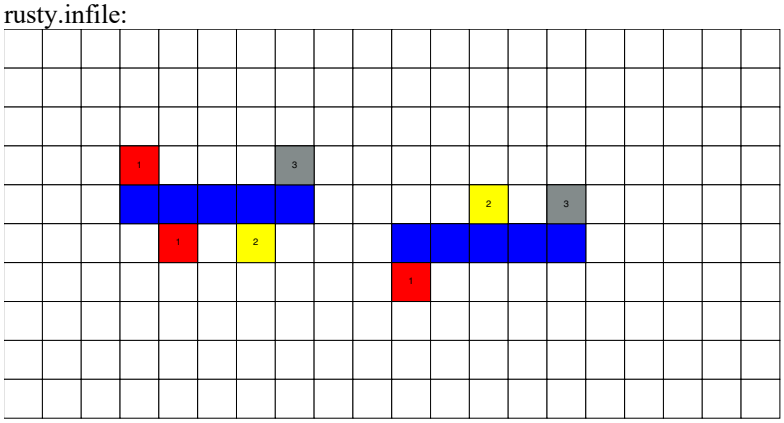
impossible.infile:



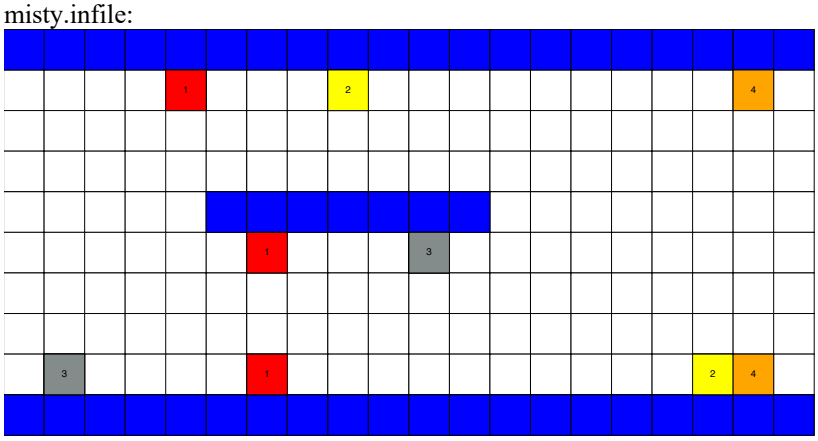
oswald.infile:



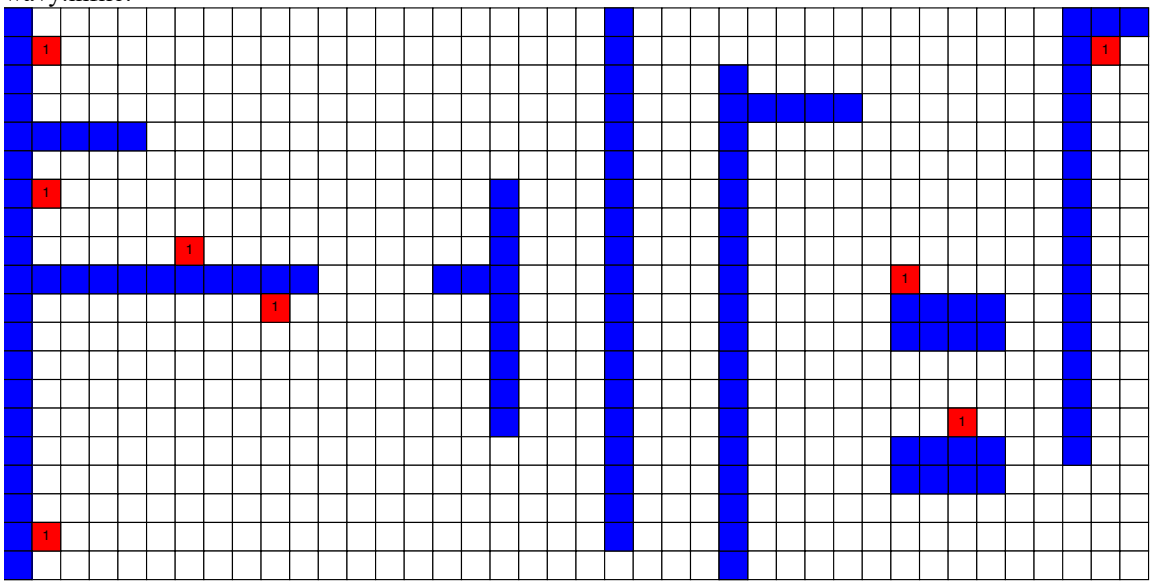
rusty.infile:



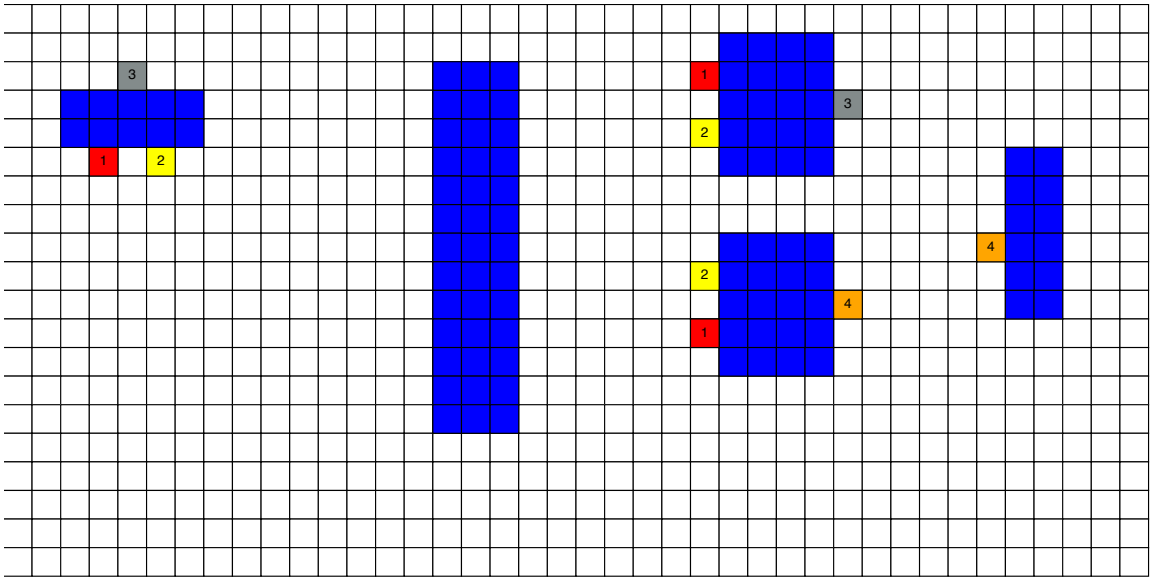
misty.infile:



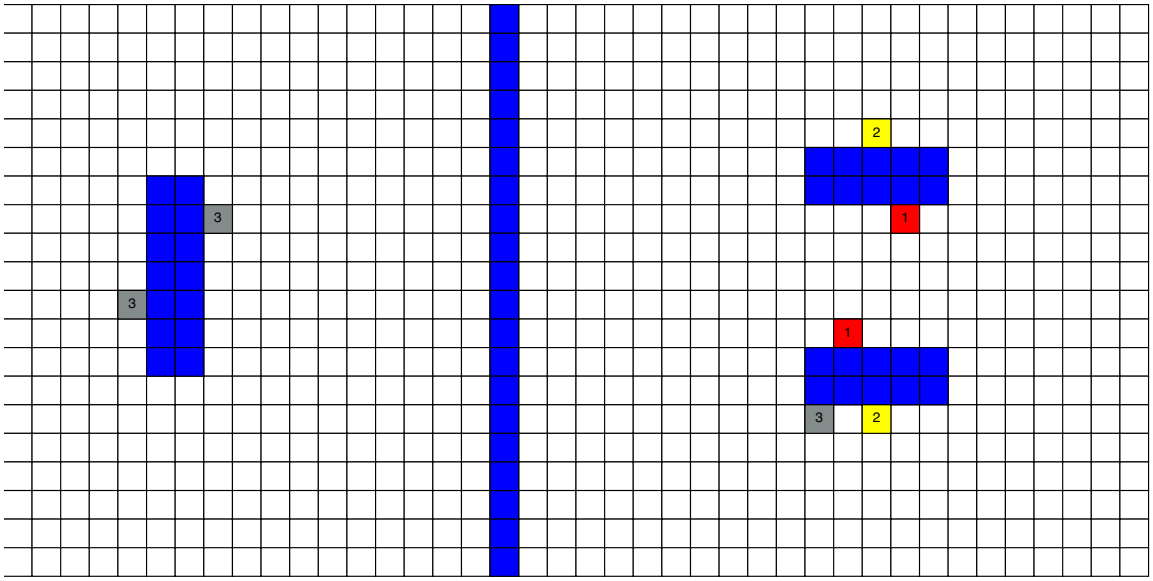
wavy.infile:



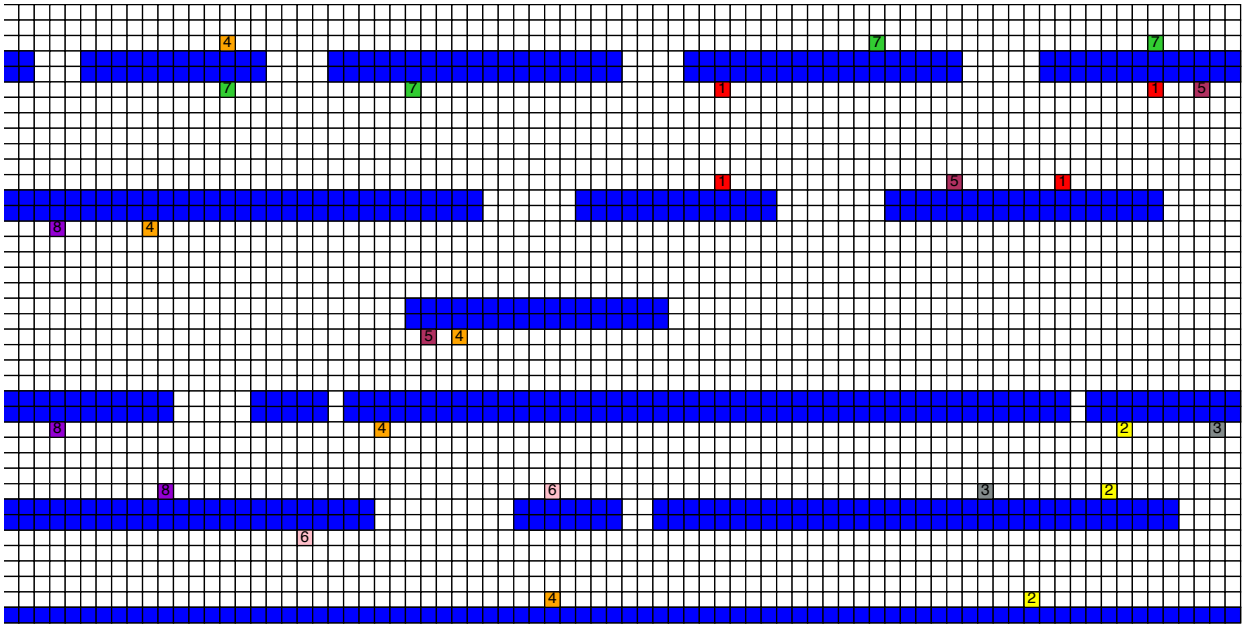
kuma.infile:



impossible2.infile:



stdcell.infile:



Appendix C: Marking Scheme for Assignment 1: Maximum Score = 20

Basic Lee-Moore Algorithm:

Score	0	2	4
Description	The algorithm does not route a single 2-point net correctly.	The implementation routes a single 2-point net correctly *most of the time*.	The implementation routes a single 2-point net correctly

A* Algorithm:

Score	0	1	2
Description	A* Algorithm not implemented, or it doesn't work properly for many benchmark circuits	The A* algorithm works *most of the time*.	The A* algorithm is implemented correctly

Multiple Sinks:

Score	0	2	4
Description	The algorithm does not appear to handle multiple sink nets well.	The algorithm handles multiple sinks in a relatively basic way.	The algorithm handles multiple sinks an in intelligent way.

Graphics:

Score	0	1	2
Description	No graphics.	The graphics only show the final solution (or a small number of intermediate solutions)	The graphics clearly shows the progress of the algorithm as it progresses.

Code Quality:

Score	0	1	2	3
Description	Code is lacking in structure and comments.	Code is of sufficient "academic code" quality, including some comments.	Code is of good "academic code" quality including extensive comments. Code is well structured.	Code is of industrial quality, including evidence of unit tests and/ or extensive system tests.

Report (maximum 3 pages):

Score	1	2	3
Description	Report is unclear or difficult to read and/or understand..	Report describes most aspects of marking scheme clearly. The English has grammar/clarity errors that would not be acceptable in a major IEEE or ACM journal or conference.	Report describes all aspects of marking scheme clearly. The English is of a professional standard that would be acceptable in a major IEEE or ACM journal or conference.

Initiative: (be sure to identify any extensions in your report):

Score	0	1	2
Description	Assignment implemented as in handout.	The implementation contains one or more straightforward extensions.	The implementation goes beyond what is described in the handout in a non-trivial way.