Assignment 3: Tiles of Monkey Island

Write a report (pdf or word) in which you answer the questions below. For each question:

1. repeat the question you are answering;
2. explain your approach;
3. describe your answer(s);
4. show (relevant) code snippets;
5. show the output of your program.

Once your report is finished, make sure your names and student numbers are on the title page, and send it to your instructor before the deadline.

Note that you are required to program in C# or C++, and must work in pairs of two. Assignments are graded with a G (good), V (sufficient) or O (insufficient).

Download the source code of "Tiles of Monkey Island" from the VLO, and open it in your IDE. Make sure you understand the code before you start writing your report.

In the (unzipped) folder you downloaded from the VLO, you'll find ten 40x30 bitmap images (*i1* to *i10*), each pixel representing a tile on a two-dimensional island. The island has walkable and non-walkable tiles. If a tile is non-walkable it means that it contains an obstacle and that you cannot include this tile in your path. The different colors in the bitmap indicate different type of tiles. The walkable tiles have different texture: road, sand, water and mountain. The table below describes their colors and costs.

Walkable type	Color	Cost	Diagonal

Road	White	10	14
Sand	Yellow	14	20
Water	Blue	20	28
Mountain	Grey	24	34

Consider the first image i1.png. It shows a game map with five (currently unconnected) islands. From a game design perspective, we are interested in placing bridges between the islands, so the player can reach other islands

Question 1)

Drawing the game map as a graph, connect the islands by bridges such that there exists a path that uses all bridges exactly once.

Hint: represent the map as a ***undirected, unweighed*** *graph.*

Question 2)

Consider the graph you have just created. Is it an Euler-graph, a Semi-Euler-graph, or neither? Explain why.

Question 3)

Is the map you have created a Hamilton-graph, a Semi-Hamilton-graph, or neither? Explain why. If it is not a Hamilton-graph, explain what bridges need to be changed to make it a Hamilton-graph.

In the next set of questions, we don't consider bridges, but rather look at the pixels as tiles. The cost of moving from tile A to tile B depends only on the cost of the type of tile B.

The start node (point) has the color red in the bitmap. The goal node (point) has the color green. The path found from the start to the end will be drawn in cyan (see: the table below).

Tile type	Color
Non-walkable	Black
Start	Red
End	Green
Node on the path	Cyan

Note that we are using the following RGB codes in the bitmaps generated with the program Paint.

Color	RGB code
White	0xffffff
Yellow	0xffff00
Blue	0x0000ff
Grey	0x808080
Black	0x000000
Red	0xff0000
Green	0x00ff00
Cyan	0x00ffff

The colors are in the standard palette of Paint.

The provided code functionality (mainly to be found in TileWorld.cs) is:

- Reading a bitmap image from an input directory that represents a tile based world and translate it to a data structure that contains information about the tile types in the tile based world.
- Saving a tile world to an output directory as a bitmap image. You can use this for saving your solution. The saved image contains the found path drawn in cyan.
- Print the information about the nodes count on screen; amount of nodes visited and the costs of the shortest path.

In the Algorithm folder, you'll find three classes that describe a search mechanism that decides what node the pathfinding algorithm should explore next: Dijkstra.cs, AStar.cs and BFS.cs.

Question 4)

Finish the code in AStar.cs so that it returns the right value (hint: take a look at Dijkstra.cs and BFS.cs)

Question 5)

Which of these three search mechanisms are informed, and which are uninformed? Explain your answer by pointing out the relevant code snippets.

Question 6)

Run the program using the twenty-one images, and fill out the results of algorithms (i.e. the lenght of the path found) using the table below. If an algorithm does not find a solution this should be indicated by a path cost of -1.

Image	A*	Dijkstra	BFS
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			

Question 7)

Choose an image where (at least) two path finding algorithms found a different path length. Explain, considering the image, why they did not find the same path.

Question 8)

The best-first search (BFS) search method currently uses a heuristic called Manhattan Distance (http://xlinux.nist.gov/dads//HTML/manhattanDistance.html). Change the code such that it uses a different heuristic.

For example, you can use the Euclidian Distance between two tiles (http://xlinux.nist.gov/dads//HTML/euclidndstnc.html). Other heuristics are also possible.

Question 9)

Repeat the experiment of question 6. Is there a difference in results for the BFS pathfinder, given that it uses a different heuristic? If yes, explain why so using the resulting images. If no, explain why the heuristic makes no difference here.

Question 10)

Create your own island map – a 40x30 png image, using the colors prescribed in the tables above - where you expect different paths found for all three algorithms. Run the pathfinding algorithms on your map, and show the results.