Name:	
Course:	CAP 4601
Semester:	Summer 2013
Assignment:	Assignment 05
Date:	26 JUN 2013

Complete the following written problems:

1. Problem Set 1 (100 Points).

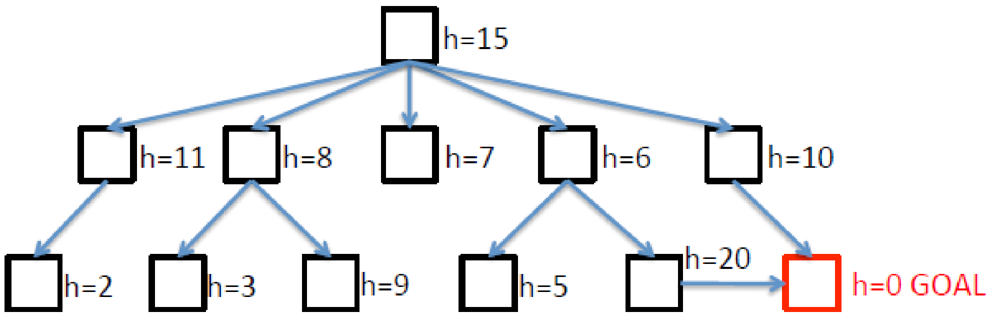
Complete the quizzes in the "Problem Set 1" unit in the Introduction to Artificial Intelligence course from Udacity:

a. Sign In to Udacity.
b. Select Introduction to Artificial Intelligence from the Course Catalog.
c. Press the "Take the Class" button
d. Take the quizzes in the "Problem Set 1" unit.

There is nothing to write or turn in for this problem. Everyone that turns in this assignment will receive these 100 points. Ensure that you pay close attention during these quizzes.

2. A* Search (100 Points).

For heuristic function h and action cost 10 (per step), enter into each node the order (1,2,3,...) when the node is expanded (=removed from queue). Start with "1" at start state at the top. Enter "0" if a node will never be expanded.



a. The node where h=15: 1
b. The node where h=11:
c. The node where h=8:
d. The node where h=7:
e. The node where h=6:
f. The node where h=10:
g. The node where h=2:
h. The node where h=3:
i. The node where h=9:
j. The node where h=5:
k. The node where h=20:
j. The node where h=0 GOAL:

l. Is this heuristic admissible? If yes, why? If no, why?

Complete the following programming problems on `linprog4.cs.fsu.edu`:

Download the ZIP file containing the directory structure and files for the following programming problem: [`assignment_05.zip`]

1. A* Search (200 Points).

Implement the A* Search algorithm in C++11 to find the shortest path between any city in the simplified road map of part of Romania on page 68 and Bucharest. In other words, find the shortest path between any city on that map and Bucharest.

Use the following code:
– `main.cpp`: The file to be studied and then edited.
– `makefile`: The makefile for `linprog4.cs.fsu.edu`.

Use the links in Week 01 to understand any C++11 code that you may be unfamiliar with.

Only edit the following section in `main.cpp`:

```
//----------------------------------------------------------------------
// Begin Your A* Algorithm Below:
// Combine the Uniform-Cost-Search function on page 84 with the
// information on A* search starting on page 93.
//----------------------------------------------------------------------

// ENTER YOUR A* SEARCH CODE HERE.

//----------------------------------------------------------------------
// End Your A* Algorithm Above.
//----------------------------------------------------------------------
```

You may use the lambda functions that I provided above this section of code or you may write your own lambda functions in this section of code.

The usage for the `main.exe` that the `makefile` produces is as follows:

```
./main.exe [-source "City"]
```

where `[-source "City"]` is optional. If `-source "City"` is not provided, then Arad is used as the source city.

The destination city is always Bucharest.

For example:
`./main.exe -source "Rimnicu Vilcea"` finds the shortest path between Rimnicu Vilcea and Bucharest.

Hint: Check your implementation using the stages in an A* search on page 96. Figure 3.24 contains all the calculations that your code should carry out. Additionally, many sites across the Internet have example implementations and pseudocode for A*. Use whatever is easiest to understand and works properly, but do not edit main.cpp outside the comments above. In other words, you must use the interface that I have provided (i.e. the container types, smart pointers, etc.).

Use `std::cout` to create output of the following format ... the following is shortest path from Arad to Bucharest:

```
Source:      Arad
Destination: Bucharest

Expanded:
             { name:           Arad, f: 366, g:   0, h: 366 }
             { name:          Sibiu, f: 393, g: 140, h: 253 }
             { name: Rimnicu Vilcea, f: 413, g: 220, h: 193 }
             { name:        Fagaras, f: 415, g: 239, h: 176 }
             { name:        Pitesti, f: 417, g: 317, h: 100 }
             { name:      Bucharest, f: 418, g: 418, h:   0 }

Solution:    Arad -> Sibiu -> Rimnicu Vilcea -> Pitesti -> Bucharest
```

Note: This output contains similar information from Figure 3.24 on page 96.

Here's another example of output ... for the shortest path from Timisoara to Bucharest:

```
Source:      Timisoara
Destination: Bucharest

Expanded:
             { name:      Timisoara, f: 329, g:   0, h: 329 }
             { name:          Lugoj, f: 355, g: 111, h: 244 }
             { name:        Mehadia, f: 422, g: 181, h: 241 }
             { name:           Arad, f: 484, g: 118, h: 366 }
             { name:        Drobeta, f: 498, g: 256, h: 242 }
             { name:          Sibiu, f: 511, g: 258, h: 253 }
             { name: Rimnicu Vilcea, f: 531, g: 338, h: 193 }
             { name:        Fagaras, f: 533, g: 357, h: 176 }
             { name:        Pitesti, f: 535, g: 435, h: 100 }
             { name:        Craiova, f: 536, g: 376, h: 160 }
             { name:      Bucharest, f: 536, g: 536, h:   0 }

Solution:    Timisoara -> Arad -> Sibiu -> Rimnicu Vilcea -> Pitesti -> Bucharest
```

HINT: Even though C++11 has a `std::priority_queue`, do not use it for this problem . . . it will be more trouble than it's worth.

After completing Assignment 05, create an `assignment_05_`*`lastname`*`.pdf` file for your written assignment and an `assignment_05_`*`lastname`*`.zip` file for your programming assignment (where *`lastname`* is your last name). Ensure that your `assignment_05_lastname.zip` retains the directory structure of the original zip file. In other words, ensure your zip file has the following directory structure:

- `/`
    - `a_star/`
        - `main.cpp`
        - `makefile`

Upload both your `assignment_05_`*`lastname`*`.pdf` file for your written assignment and your `assignment_05_`*`lastname`*`.zip` file for your programming assignment to the Assignment 05 location on the BlackBoard site: https://campus.fsu.edu.