Introduction With suitable abstractions planning a path for a mobile robot can be converted into a search problem. Begin by abstracting the environment into a 2D grid of square “cells”. The robot state can be represented by a [x,y] pair, with [0,0] being the top-left cell of the grid. Movements of the robot are possible at any time either UP, DOWN, LEFT RIGHT (denoted U, D, L, R) unless the neighbouring cell is occupied. The UP action moves to the cell immediately above (if possible) by changing [x,y] to [x,y-1]. Likewise RIGHT changes the current state from [x,y] to [x+1,y], and so on. Given a start state, the goal of the robot is to reach a new (user specified) state [X*, Y*], where this must be an unoccupied cell in the grid. An action that would take the robot outside the grid or into an occupied cell results in no change to the current state. Your task is to write a program that will read in an environment, a start state and a goal state, and conduct a search to find a path between start and goal. You may implement any of the search algorithms discussed in lectures. Your output should be in the form of a space-separated list of actions, such as U R R D D L L (more specific detail is given below). Test data will be provided on the course pages along with this Assignment description. Different data will be present in the submission system and your search program will be evaluated against these unseen data for its completeness and optimality. You must write the program yourself in either C, C++, Java or Python. You may freely use libarries or packages for data structures (e.g. for queues or trees or other structures you deem necessary), but if you use a library package or language function call for doing the search itself, you will be limited to 50% of the available marks. If there is evidence you have simply copied code from the web, you will be awarded no marks and referred for plagiarism. The assessment script will attempt to work out what language yo have used by looking at file extensions so do not mix .py, .c, .cc, .java files in the same repository. If you are submitting C/C++ you must also submit a makefile and the script will attempt to compile using the command make. If you submit java the script will attempt compilation using javac robotplanner.java, and for python there is no compilation. 1 The program must accept 5 arguments from the command-line: a file (which contains the environment); and 4 integers specifying start state and goal state, eg: ./robotplanner env.txt 0 2 4 1 would read the environment specification from the file env.txt and should plan a path from [0 2] to goal state [4 1]. The command above is run by the script if your program is C/C++. For java the command is java robotplanner env.txt 0 2 4 1 and for python it is python robotplanner.py env.txt 0 2 4 1 The search algorithm you use is deliberately not specified. You can use any search algorithm that you believe will produce a solution. But take note that undergraduates need to show their implementation produces a correct path. Postgraduates have in addition to produce an optimal path. The output of your program must strictly comply with the specs below otherwise the automarking script will not be able to parse your answer, and will therefore possibly not award all the marks available. Submission and Assessment Submit your program on the Computer Science Web Submission System. Undergraduates will be assessed on whether they produce a valid solution. There are 18 tests on 3 different grids (sized 5x5, 20x15 and 9x9). If a path is not possible and your program correctly diagnoses this, the test is worth 4 marks. If the path is possible and you find one valid path, you get 6 marks. Maximum available is 100. If your program produces something that gives 0 marks please make sure you still submit since I will attempt to award marks for failed programs. the better your code is documented, the more likely it is you will awarded marks for submitting a failed attempt.