Module Code
COM00037H

BSc, BEng Degrees Examination 2021—22

DEPARTMENT OF COMPUTER SCIENCE

EVAC

Module Component 1

Open Individual Assessment

Release to Students: Wednesday 16th February, 2022 Midday (Spr/6/Wed)

Hand-in from Students: Wednesday 16th March, 2022 Midday (Spr/10/Wed)

Feedback and Marks Released: Wednesday 20th April, 2022 (Sum/1/Wed)

All students should submit their answers through the electronic submission system: http://www.cs.york.ac.uk/student/assessment/submit/ . An assessment that has been submitted after the deadline (above) will be marked initially as if it had been handed in on time, but the Board of Examiners will normally apply a lateness penalty.

Your attention is drawn to the section about Academic Misconduct in your Departmental Handbook: https://www.cs.york.ac.uk/student/handbook/.

Any queries on this assessment should be addressed by email to Prof Dan Franks: Daniel.Franks@york.ac.uk. Answers that apply to all students will be posted on the VLE.

Rubric

Your report **must** respect the following structure and criteria:

- Maximum page limit of **5 pages** (excluding references & cover page)
- Sans Serif Arial font size 12, with headings no larger than size 14
- Use the main section headings: Methods, Results, Conclusions, References
    - Subheadings can be used within these main headings
- The report starts on the second page after the coversheet
- Pages numbered with the X of Y format
- Add 'End of Paper' at the end of the assessment
- Single column
- Minimum of 2cm margins on all four sides

Parts of answers that exceed the page limit or font criteria are not marked

The Problem: Evolve a Player for the Video Game Snake

Snake is a classic video game where the player controls a snake moving around a grid. In each time-step the snake moves forward in the direction it is facing. The aim of the game is for the snake to collect food that appears around the grid. Each food item increases the score by one. The game is over if the snake's head collides with its own body, collides with the wall around the grid, or if it has been too long since the snake last ate. Each food item eaten makes the snake longer, and thus the game becomes increasingly difficult.

Your Task

Design and implement an evolutionary algorithm in Python to create an agent to play this game. You should attempt different solutions based on a principled and inventive approach to the design and implementation of your algorithm.

Documents that you will submit:

a) A written report (see rubric for criteria and formatting details)
b) The Python code for your evolutionary algorithm

The Rules of Implementation

- You *must not* manually add any further movement options to the snake. It can only change direction using the provided commands (e.g. you cannot add relative movement directions).
- You *must not* hand-code solutions for the snake. They must be found by evolutionary computation.
- You *may* add any environmental sensing functions that you like to the snake (from the full state of the grid, down to local sensing).

The Files Provided

Go to the exam page on the EVAC VLE and download the two files provided. These are:

a) snakePlay.py
b) snakeProblem.py

File (a) will allow you to familiarize yourself with the game by playing it. Do this by executing the game from the terminal with: `python snakePlay.py`

File (b) code on which you should build your algorithm. Snake movement commands are already included, along with a simple example of how to sense the environment.

What to Include in the Report

You are expected to show a principled and inventive approach to the above investigation and communicate your findings effectively and methodically. You should write a reproducible Methods section, an appropriately analysed and presented Results section, and a short Conclusions section. You should include all key figures and statistics in the main

report. Any supplementary figures and stats can be supplied in the appendix but will be treated as supplementary rather than core results.

General Marking Overview for Students

Code [5 marks]

Suitable code and solution for the evolutionary algorithm using Python and a good final solution and runs on a test game when compiled

Methods [45 marks]

- Reproducible methods / representation and design [12 marks]
- Quality of the design of the algorithm (e.g. consideration and choice of representation, fitness assessment, other details [13 marks]
- Additional adjustments, investigation or inventive tweaks on how to improve evolution [12 marks]
- Uses principles taught across the module (or beyond) [8 marks]

Results [45 marks]

Evaluate your snake(s) and variants of your algorithms and use appropriate summary statistics and plots in your reporting.

- Express what will be tested /solution quality [6 marks]
- Appropriate investigation visualization and reporting of results [15 marks]
- Appropriate interpretation and discussion of results [10 marks]
- Test between variants of algorithm and informative tests between conditions [14 marks]

Conclusions [5 marks]

Summarizes findings well and highlights key points. Demonstrates critical ability. Provides thoughtful suggestions for future work.

END OF PAPER