TEESSIDE UNIVERSITY
SCHOOL OF COMPUTING

ICA SPECIFICATION

This form should be attached to the front of the ICA specification and include all details.

Module Title: **Artificial Intelligence Programming**	**Module Leader**: S.Lynch
	Module Code: VIS2067-N
Assignment Title: **Planning**	**Deadline Date:** 16 May 2011
	Deadline Time: 16.30
	Submission Method: Student Advice & Support Centre ■ ~~Online~~ ☐

Student Advice & Support Centre Submission Notes:

- All work (including CDs etc) needs to be secured in a plastic envelope or a folder and clearly marked with the student name, number and module title.
- An Assignment Front Sheet should be fully completed before the work is submitted to the Student Advice & Support Centre.
- When an extension has been granted, a fully completed and signed Extension form must be submitted to the Student Advice & Support Centre together with your work.

ONLINE Submission Notes:

- Please follow carefully the instructions given on the Assignment Specification
- When an extension has been granted, a fully completed and signed Extension form must be submitted to the Student Advice & Support Centre as soon as it had been signed.

FULL DETAILS OF THE ASSIGNMENT ARE ATTACHED
INCLUDING MARKING & GRADING CRITERIA

Assignment Aim

The aim of this assignment is to test the students' ability to design and develop solutions to problem based learning exercises. The students will also be required to provide a report (1000-1500 words), written and presented to a professional standard, to discuss and critically evaluate the development process of their solutions.

The assignment is to be undertaken as group work. Group sizes will be between 2 and 4 students.

Learning Outcomes

On successful completion of this module, the student will be able to:

1. Demonstrate how to program a simple agent (robotic agent or game non-player character) to act upon its environment.
2. Evaluate different approaches to search and their relative benefits in different situations.
3. Create small scale AI systems to solve problems, plan sequences of actions or play games.
4. Construct simple knowledge representational schemes and the inference mechanisms that support them.
5. Use Common Lisp to build symbolic representations (rules, knowledge, planning operators, etc) for use with a selection of inference mechanisms.
6. Research appropriate resources.
7. Produce documentation for software design.

Marking Scheme

This assignment accounts for **75%** of the overall mark for the assessment of the module, where the remaining **25%** are allocated to the assessment of the presentations which were undertaken during the course of the year during computer lab sessions.

Hand in

You are expected to submit both the report (1000-1500 words) and the Lisp code in electronic format on disk (CD or DVD) with the report in Microsoft Word or Adobe Acrobat format. You are also expected to provide a single printed copy of the report and all program code. Program code sample should include the following...

1. static world definition
2. operators (grouped into levels)
3. code to coordinate operators working at different levels
4. sample start states and goal states (at least 3 of each)
5. sample test runs (with explanations)

Brief

General Problem Solving using STRIPS-like operators and the search mechanisms provided on www.agent-domain.org.

The main objectives of the assignment are:

1. use sets of tuples to specify domain state descriptions
2. design and implement STRIPS-like operators
3. investigate the limitations of simple-search strategies

In this assignment, you will be solving a standard planning-like problem for which you will need to define the context for a virtual world (its characteristics and properties) within which a one or more agents carry out state-changing activities. You will develop the solution by describing the domain using sets of tuples and also STRIPS-like operators to be applied on the described domain.

You should design two sets of operators to work at two different levels of abstraction. The *higher* level operators will be used to formulate the main steps of a high level plan, the *lower* level operators will be used to plan sequences of state-changing activities for each step in the higher level plan.

In order to test your operators, you are required to set several problems to be solved defined as pairs of start and goal states. These start and goal states should be specified as sets of tuples.

You are not required to implement your own search mechanisms but you should use the following Lisp tools: *matcher*, *utils* and *ops-search* which are available from the module's webpage. You may also use/write any Lisp functions as necessary.

Note that the context and scale of the domain problem you are setting should be agreed with the module tutors prior to any further development.

An example

A virtual world environment contains 9 rooms with doors between them See diagram below. Doors may be open or closed but may not be locked. There is a cupboard and a table in the kitchen and a bed in the bedrooms. There is a robot in the world. The robot has some kind of arm which allows it to hold (at most) one item, it also has some device which allows it to climb on furniture like tables and chairs.

After drawing a the sketch of the world (above) it should be specified as 2 sets of tuples describing (i) static/unchanging information and (ii) changeable state information. Operators can then be specified to work on this information.

For example, high-level operators can be specified (i) to move from one room to any other room (in one step regardless of the need to open doors, etc and move through intervening rooms) (ii) to pick-up & set-down objects (as long as they are in the same room as the robot but without needing to be next to that object).

Lower-level operators can be specified to deal with activities like opening & closing doors, moving between rooms and the tuple changing activities required to change world states.

Sample start and goal states define example tasks which can be solved by plans (plans are specified in terms of sequences of operator applications). An example of a task could be to move a number of items from the bedrooms to the kitchen.

Assessment criteria

You are expected to submit both the report (1000-1500 words) and the Lisp code in electronic format on disk (CD or DVD) with the report in Microsoft Word or Adobe Acrobat format. You are also expected to provide a single printed copy of the report and all program code. Program code sample should include the following...

1. report
2. static world definition
3. operators (grouped into levels)
4. code to coordinate operators working at different levels
5. sample start states and goal states (at least 3 of each)
6. sample test runs (with explanations)

The following table is intended to give you further guidance on the submission requirements. Note that the table is not intended to provide a checklist and should be used to supplement the requirements given in the task specifications.

requirement	**Pass (40-59)**	**Good (60-69)**	**Excellent (70+)**
(1) report	demonstrated understanding of main issues & objectives; solution is presented; results are assessed	good understanding of the problem, appreciation of main issues & objectives; clearly presented solution; results are explained	full (but concise) explanation of the problem domain and the objectives of the task, clear insight into main challenges & compromises; well argued solution; accurate critique of results
(2, 3) world definition & operators	world is unambiguously described by tuples; an adequate set of operators are defined; operators are mostly consistent with world domain	world is well described by suitable tuples; most required operators are defined; operators are adequately specified and consistent with world domain	world is accurately & concisely described by well chosen sets of tuples; complete & well chosen operators; operators well-specified and consistent with world domain
(4) coordination of operators	the code is reasonably constructed; it works for the examples presented;	the code is reasonably constructed & readable; it works & may be further developed;	the code is well constructed & easy to understand; it works well & well suited to further development;
(5, 6) sample tests	results adequate to highlight main behaviour; presentation reasonably clear;	results clearly show main behaviour; chosen tests are adequate to support further discussion;	well presented results which show expected behaviour and highlight any other interesting features; choice of tests suited to further analysis;