Matthew Abraham

815009613

COMP 3950

Assignment 2 (4 Credit)

Part 1 & 2

Fig 6.PL

```
Call:run
 Call:r5
 Call:fact(l)
 Fail:fact(l)
 Redo:r5
 Call:r4
 Call:fact(c)
 Exit:fact(c)
 Call:assert(fact(l))
 Exit:assert(fact(l))
 Exit:r4
 Exit:r5
 Call:r4
 Call:fact(c)
 Exit:fact(c)
 Call:assert(fact(l))
 Exit:assert(fact(l))
 Exit:r4
 Call:r3
 Call:fact(a)
 Exit:fact(a)
 Call:assert(fact(x))
 Exit:assert(fact(x))
 Exit:r3
 Call:r2
 Call:fact(x)
 Exit:fact(x)
 Call:fact(b)
 Exit:fact(b)
 Call:fact(e)
 Exit:fact(e)
 Call:assert(fact(y))
 Exit:assert(fact(y))
```

The program checks each fact before executing the respective rule. If the fact does not exists, it goes to the rule that asserts that fact. At the beginning, r5 calls the fact(l) which has not been created yet. It then fires r4 to attempt to assert that fact. The execution of the program continues in the procedure r5->r4->r3->r2->r1.

CLIPS pulls the first rule from the bottom of the file which can be fired.

Fig 6.CLP

```
FIRE    1 r4: f-3
==> f-6     (L)
FIRE    2 r3: f-1
==> f-7     (X)
FIRE    3 r2: f-7,f-2,f-5
==> f-8     (Y)
FIRE    4 r1: f-8,f-4
==> f-9     (Z)
```

1. R4 (since r5 is lower down in the file but the facts L and M do not exist in the knowledge base at that moment) and creates the fact L.
2. The next one that can be fired (moving towards the top of the file) is r3 which asserts fact X.
3. After this, r2 is fired, and the facts X, B, E exist which asserts the fact Y.
4. Lastly, at the top of the file, r1 is fired and asserts the fact Z.

The knowledgebase starts off with 5 facts and ends with 9 facts.

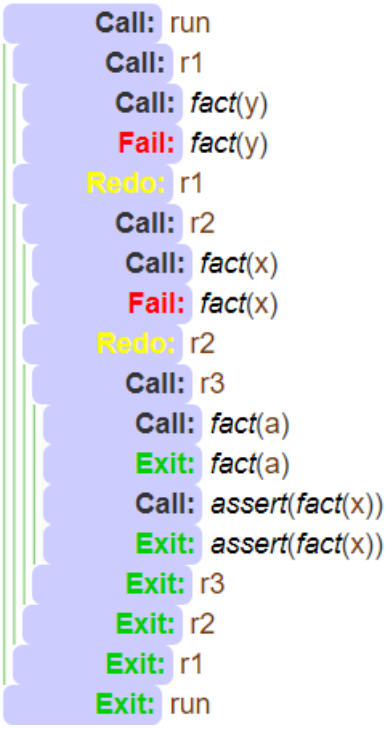
Fig 5.CLP

```
CLIPS> (run)
FIRE    1 r3: f-1
==> f-6     (X)
FIRE    2 r2: f-6,f-2,f-5
==> f-7     (Y)
FIRE    3 r1: f-7,f-4
==> f-8     (Z)
```

1. R3 is fired first since fact A exists, asserts fact X.
2. R2 is fired next, facts X, B, E exist, asserts fact Y.
3. R1 is fired, facts Y, D exist, asserts fact Z.

Knowledgebase starts off with 5 facts and ends with 8 facts.

Fig 5.PL



The main function calls upon the first rule, r1. At this state of execution, the fact Y does not exist, so it attempts to solve by calling r2. At this state, the fact X does not exist, so it attempts to solve by calling r3. At this state, fact A exists, asserts X, returning to r2, which then asserts Y, returning to r1, which then asserts Z.

This is an example of backward chaining by solving for specific subgoals within the knowledge base.

The knowledge base starts off with 5 facts and ends with 8 facts.

Part 3

Assumptions made for this section were that the accepted inputs for ENVIRONMENT were (papers, manuals, documents, textbooks, pictures, illustrations, photographs, diagrams, machines, buildings, tools, numbers, formulas, programs) and the accepted inputs for JOB were (lecturing, advising, counselling, building, repairing, troubleshooting, writing, typing, drawing, evaluating, reasoning, investigating), and the accepted inputs for FEEDBACK were (yes, no).

```
CLIPS> (run)
FIRE    1 ask-environment: *
What is the environment?
manuals
==> f-1     (environment manuals)
FIRE    2 stimulus-situation-verbal: f-1
==> f-2     (stimulus-situation verbal)
FIRE    3 ask-job: *
What is your job?
lecturing
==> f-3     (job lecturing)
FIRE    4 stimulus-response-oral: f-3
==> f-4     (stimulus-response oral)
FIRE    5 ask-feedback: *
Do you need feedback? (yes/no)
yes
==> f-5     (feedback yes)
FIRE    6 medium-roleplay: f-2,f-4,f-5
==> f-6     (medium roleplay)
CLIPS>
```

CLIPS

The fact ENVIRONMENT VERBAL is added to the knowledgebase, and fires the rule STIMULUS-SITUATION-VERBAL.

The fact JOB LECTURING is added to the knowledgebase, and fires the rule STIMULUS-RESPONSE-ORAL.

This fires the rule MEDIUM-ROLEPLAY as the result of the user's input.

```
Call:prompt(textbooks, lecturing, yes, _4704)
 Call:assert(environment(textbooks))
 Exit:assert(environment(textbooks))
 Call:assert(job(lecturing))
 Exit:assert(job(lecturing))
 Call:assert(feedback(yes))
 Exit:assert(feedback(yes))
 Call:seek_medium(_4704)
 Call:medium_roleplay
 Call:feedback(yes)
 Exit:feedback(yes)
 Call:situation(verbal)
 Fail:situation(verbal)
 Redo:medium_roleplay
 Call:situation_verbal
 Call:environment(papers)
 Fail:environment(papers)
 Redo:situation_verbal
 Call:environment(manuals)
 Fail:environment(manuals)
 Redo:situation_verbal
 Call:environment(documents)
 Fail:environment(documents)
 Redo:situation_verbal
 Call:environment(textbooks)
 Exit:environment(textbooks)
 Exit:situation_verbal
 Call:response_oral
 Call:job(lecturing)
 Exit:job(lecturing)
```

PROLOG

The fact environment(textbooks) is added to the knowledgebase.

The fact job(lecturing) is added to the knowledgebase.

The fact feedback(yes) is added to the knowledgebase.

The rule medium_roleplay is fired and solves subproblem of situation(verbal).

The rule situation_verbal is fired and adds the fact situation(verbal) to the knowledgebase since environment(textbooks) is true.

The rule response_oral is fired and adds the fact response(oral) to the knowledgebase since job(lecturing) is true.

Medium is set to “Roleplay” and execution reaches a halt.