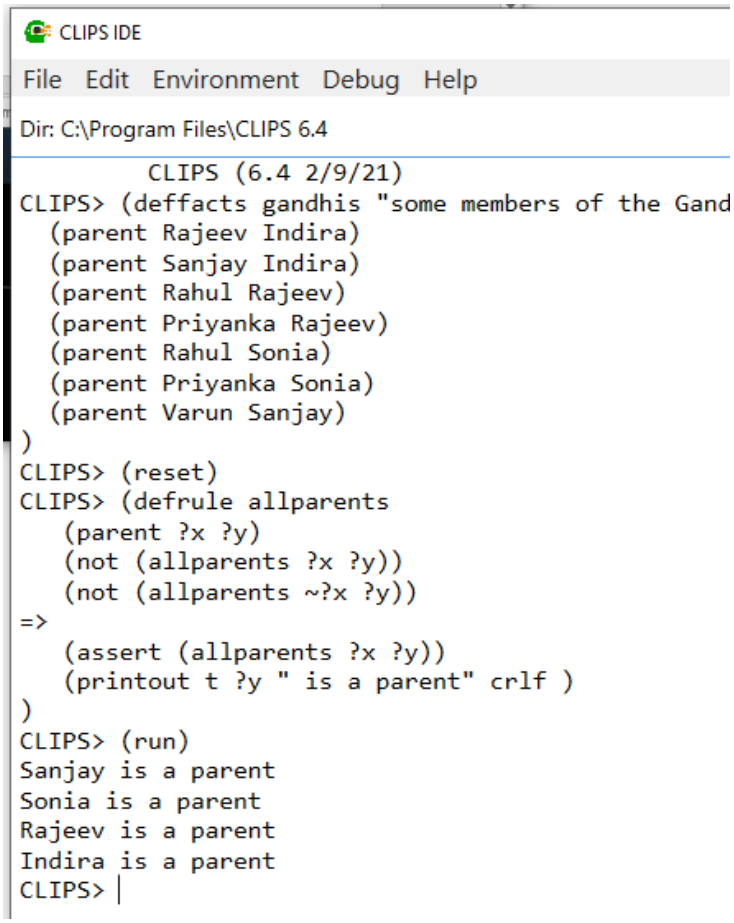
Homework 2

FULYA KOCAMAN
CWID: 803023878

1. The attached `family.clp` CLIPS program describes a set of parent-child pairs and rules to identify siblings. Remember to run `(reset)` to load the facts.

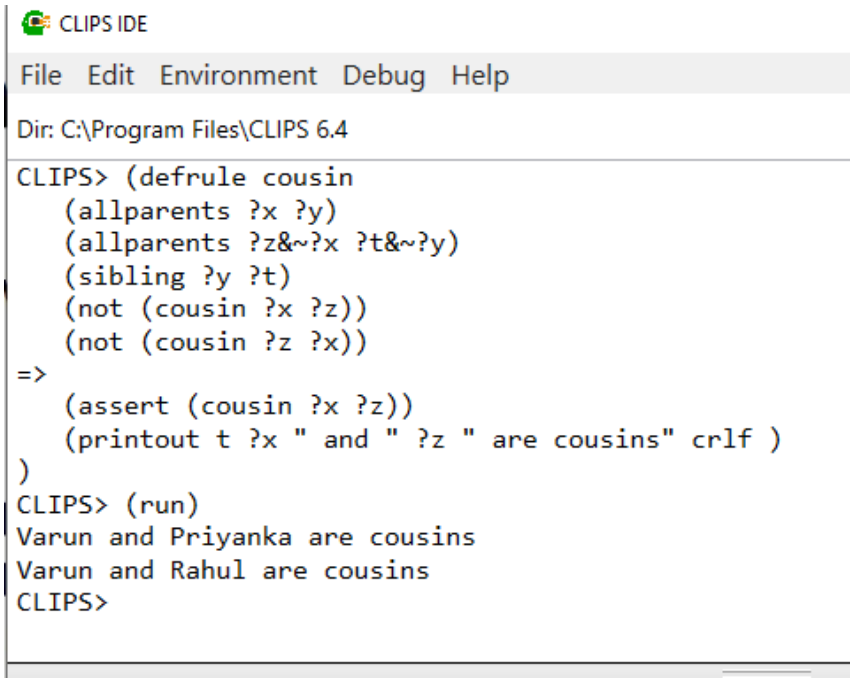
   a. Add one new rule to family.clp to print a list of all people who are parents. Do not add/remove facts. [Show rule and output of the program]

```
(defrule allparents

  (parent ?x ?y)

  (not (allparents ?x ?y))

  (not (allparents ~?x ?y))

=>

  (assert (allparents ?x ?y))

  (printout t ?y " is a parent" crlf )

)
```



   b. Add one new rule to family.clp to print a list of all pairs of persons who are cousins *and* assert new facts of the form `(cousin Varun Rahul)`. Two persons are cousins if their parents are siblings. You do not have to prevent duplicate pairs. [Show rule and output of the program]

```
(defrule cousin

  (allparents ?x ?y)

  (allparents ?z&~?x ?t&~?y)

  (sibling ?y ?t)

  (not (cousin ?x ?z))

  (not (cousin ?z ?x))

=>

  (assert (cousin ?x ?z))

  (printout t ?x " and " ?z " are
cousins" crlf )

)
```



Please note that in order to print all pairs of cousins, I had to adjust my allparents rule from part (a). In part (a) I added (not (allparents ~?x ?y)) not to print the same parent twice coming from the fact of a sibling, but if I had kept the allparents rule as is from part (a), I would have missed the second sibling to match with their cousins. So, in part (b) I removed the line (not (allparents ~?x ?y)) from the allparents rule to print all pairs cousins.

2. The attached `cars.clp` CLIPS program describes a list of cars (brand, price, color). The program asks the user to enter an age and then executes a rule to recommend a car – if the age is less than 25, then recommend cars that cost less than $30,000.

   a. Modify the rule such that the recommendations for a person younger than 25 years of age are cars that cost less than $30,000 *and red in color*. [Show modified rule and output of the program]

```
(defrule recommend-car

  (age ?a)

  (test (<= ?a 25))

  (car ?car ?price Red)
```

```
  (test (<= ?price 30000))

=>

  (printout t " Recommeding " ?car
crlf )

)
```

```
CLIPS IDE
File  Edit  Environment  Debug  Help
Dir: C:\Program Files\CLIPS 6.4
CLIPS> (defrule get-age
  (not (age ?))
=>
  (printout t "What is the person's age? " )
  (assert (age =(read)))  ; Read answer and add it as a fact
)

(defrule recommend-car
  (age ?a)
  (test (<= ?a 25)) ; test if age <= 25
  (car ?car ?price Red)
  (test (<= ?price 30000)) ; test if price <= $30,000
=>
  (printout t " Recommeding " ?car crlf )
)
CLIPS> (run)
What is the person's age? 18
 Recommeding Ford
 Recommeding Honda
CLIPS>
```

b. Add a new rule that recommends for a person older than 25 years a white car. [Show new rule and output of the program]

```
(defrule recommend-car2

  (age ?a)

  (test (> ?a 25)) ; test if age > 25

  (car ?car ?price White)

=>

  (printout t " Recommeding " ?car
crlf )

)
```

```
CLIPS IDE
File  Edit  Environment  Debug  Help
Dir: C:\Program Files\CLIPS 6.4
CLIPS> (defrule get-age
  (not (age ?))
=>
  (printout t "What is the person's age? " )
  (assert (age =(read)))  ; Read answer and add it as a fact
)

CLIPS>
(defrule recommend-car2
  (age ?a)
  (test (> ?a 25)) ; test if age > 25
  (car ?car ?price White)
=>
  (printout t " Recommeding " ?car crlf )
)
CLIPS> (run)
What is the person's age? 30
 Recommeding Acura
 Recommeding Toyota
CLIPS>
```

3. The attached `oldest.clp` CLIPS program was shown in class and prints the oldest age in a set of facts of the form `(person (name Rajeev) (age 46))`.

   a. Modify the rules such that the program prints the name of the oldest person along with their age (i.e., "Indira is the oldest person with age 64"). [Show modified rules and output of the program]

```
(defrule oldest-start
  (person (name ?name) (age ?age))
  (not (largest ?max))
  (not (oldestName  ?oldName))
=>
  (assert (oldestName ?name) (largest ?age))
)
(defrule oldest
  (person (name ?name) (age ?age))
  ?f1 <- (oldestName ?oldName)
  ?f2 <- (largest ?max)
  (test (> ?age ?max))
=>
  (assert (oldestName ?name))  ; the oldest so far
  (retract ?f1)
  (assert (largest ?age))  ; the largest so far
  (retract ?f2)
)
(defrule oldest-print
  (declare (salience -1))
  (largest ?max)
  (oldestName  ?oldName)
=>
  (printout t ?oldName " is the oldest person with age " ?max crlf))     ; Output is in the next page
```

CLIPS IDE

File Edit Environment Debug Help

Dir: C:\Program Files\CLIPS 6.4

```
CLIPS> (defrule oldest-start
  (person (name ?name) (age ?age))
  (not (largest ?max))
  (not (oldestName  ?oldName))
=>
  (assert
(oldestName ?name) (largest ?age))
)
  CLIPS> (defrule oldest-start
  (person (name ?name) (age ?age))
  (not (largest ?max))
  (not (oldestName  ?oldName))
=>
  (assert (oldestName ?name) (largest ?age))
)

CLIPS> (run)
CLIPS> (defrule oldest
  (person (name ?name) (age ?age))
  ?f1 <- (oldestName ?oldName)
  ?f2 <- (largest ?max)
  (test (> ?age ?max))
=>
  (assert (oldestName ?name))  ; the oldest so far
  (retract ?f1)
  (assert (largest ?age))  ; the largest so far
  (retract ?f2)
)
CLIPS> (run)
CLIPS> (defrule oldest-print
  (declare (salience -1))
  (largest ?max)
  (oldestName  ?oldName)
=>
  (printout t ?oldName " is the oldest person with age " ?max crlf)
)
CLIPS> (run)
Indira is the oldest person with age 64
CLIPS>
```