COMP1811 – Scheme Project Report

Name	Mercy Nwabueze-nwoji	SID	001155347
Partner's name	Shuaib Mahamud	SID	001153094

1. Software Design

Briefly describe the design of your coursework software and the strategy you took for solving it. – e.g. did you choose either recursion or high-order programming and why…

We had used recursion for the coding because we needed a function that was able to call itself. For the design of our coursework we had to break it into . The first part of the coursework was creating the cards which was going to be meant for solitaire. The first course of plan is to understand the base code which is given. After doing that, you need to write down all of the cards which is 1-7 or if it has a face which is jack,queen and the king and finally the suit numeral which is hearts,spade,club and diamond.Now you have to write down all of the combinations which are available for suite numerals leading to you having 40 cards. Finally define a function to use an arithmetic contrast,a quantity and a list of cards that, when given the comparison,number, and list yields the pairs of cards are much less,larger,or fair and equal to that number.

2. CODE LISTING

Provide a complete listing of the entire Scheme code you developed. Make sure your code is well commented, specially stressing informally the contracts for parameters on every symbol you may define. The code listed here must match that uploaded to Moodle. Please copy and paste the actual code – no screenshots please! Make it easy for the tutor to read. Marks WILL BE DEDUCTED if screenshots are included. Add explanatory narrative if necessary – though your in-code comments should be clear enough.

2.1 FUNCTION 0 ...

```
(define (card? val)
  (if (equal? card? val) #t

  (and

   (pair? val)

   (list? (member (car val) '(1 2 3 4 5 6 7 #\J #\K #\Q)))

   (list? (member (cdr val) '(#\D #\S #\H #\C))

          ))))
```

2.2 FUNCTION1 ...

```
(DEFINE (SUITE CARD)
   (cdr card))


(define (numeral card)

   (car card))
```

2.3 FUNCTION2

```
(DEFINE (FACE? CARD)

(if (or (equal? (car card) #\J) (equal? (car card) #\Q) (equal? (car card) #\K))
                                        #t
                                        #f))
```

2.4 FUNCTION4

```
(DEFINE (CARD->STRING VAL)
  (string-append
   (if
     (face? val) (string (numeral val))
      (number->string (numeral val)))
   (string (cdr val)
           )))
```

2.5 FUNCTION5

```
(DEFINE (CARD->STRING VAL)
  (string-append

   (if

     (face? val) (string (numeral val))

      (number->string (numeral val)))

   (string (cdr val)

           )))
```

2.6 FUNCTION6

```
;;(REQUIRE RACKET/TRACE)
(define (value card)

  (if (number? (car card))

      (and (>= (car card) 1) (<= (car card) 7) number? (car card))

           (if (char? (car card))

               (or (equal? (car card) #\S) (equal? (car card) #\H) (equal?
(car card) #\C) (equal? (car card) #\D) 0.5))))
```

2.7 FUNCTION7

```
;;USING THE FUNCTION CARD? ABOVE, DECLARE A FUNCTION DECK? WHICH, PROVIDED A LIST OF
    VALUES, RETURNS #T IF IT IS
;;a valid deck or #f otherwise

(define (deck? val)

  (if (null? val)#t

  (and (or (card? (car val))) (card? (car (cdr val))))))
```

2.8 FUNCTION8

```
(DEFINE (VALUEOF ROUND)
  (if (null? round) 0

          (+ (value (car round)) (valueOf (cdr round)))))

;;(trace valueOf)


;; hard

(define (do-suite val) (list (cons 1 val) (cons 2 val)

                             (cons 3 val) (cons 4 val)

                             (cons 5 val) (cons 5 val)

                             (cons 7 val) (cons #\K val)

                             (cons #\Q val) (cons #\J val)))
```

2.9 FUNCTION9

```
(DEFINE DECK (LIST (CONS 1 #\H) (cons 2 #\H)
                    (cons 3 #\H) (cons 4 #\H)

                    (cons 5 #\H) (cons 6 #\H)

                    (cons 7 #\H) (cons #\J #\H)

                    (cons #\Q #\H) (cons #\K #\H)

                    (cons 1 #\S) (cons 2 #\S)

                    (cons 3 #\S) (cons 4 #\S)

                    (cons 5 #\S) (cons 6 #\S)

                    (cons 7 #\S) (cons #\J #\S)

                    (cons #\Q #\S) (cons #\K #\S)

                    (cons 1 #\C) (cons 2 #\C)

                    (cons 3 #\C) (cons 4 #\C)

                    (cons 5 #\C) (cons 6 #\C)

                    (cons 7 #\C) (cons #\J #\C)
```

```
                    (cons #\Q #\C) (cons #\K #\C)
                    (cons 1 #\D) (cons 2 #\D)
                    (cons 3 #\D) (cons 4 #\D)
                    (cons 5 #\D) (cons 6 #\D)
                    (cons 7 #\D) (cons #\J #\D)
                    (cons #\Q #\D) (cons #\K #\D)))

(define (deck->strings deck)
  (map card->string deck))
```

2.10 FUNCTION 10

```
(DEFINE (PROBABILITY COMP NUM DECK)
  (cond
    [(empty? deck) 0]
    [(comp (value (car deck)) num)
     (+ 1 (probability comp num (rest deck)))]
    [else
     (probability comp num (rest deck))]))

(define cheat #f)
```

2.11 …

…

3. RESULTS – OUTPUT OBTAINED

Provide screenshots that demonstrate the results generated by running your code. That is show the output obtained in the REPL when calling your functions. Alternatively, you may simply cut and paste from the REPL.

3.1 TASK-1

```
Welcome to DrRacket, version 8.3 [cs].
Language: Pretty Big; memory limit: 128 MB.
> (card? "3 of diamonds")
#f
> (card? 3)
#f
> (card? (cons 3 #\D))
#t
> (card? (cons 9 #\D))
#f
>
```

...

3.2

Suite and numeral

```
> (suite (cons 3 #\D)) (suite (cons #\J #\D))
#\D
#\D
> (numeral (cons 3 #\D)) (numeral (cons #\J #\D))
3
#\J
>
```

3.3

Face

```
> (face? (cons 3 #\D)) (face? (cons #\K #\D))
#f
#t
> (face? (cons #\Q #\D)) (face? (cons 4 #\S))
#t
#f
>
```

3.4

Value function

```
> (value (cons 3 #\S)) (value (cons #\J #\H))
3
0.5
>  (value (cons 5 #\S)) (value (cons #\Q #\S))
5
0.5
>
```

3.5

card→string function

```
> (card->string (cons 3 #\H)) (card->string (cons 2 #\D))
"3H"
"2D"
> (card->string (cons 6 #\S)) (card->string (cons 4 #\C))
"6S"
"4C"
>
```

3.6

deck with a question mark(deck?)

```
> (deck? (list (cons 8 #\J) (cons 5 #\D)))
#f
> (deck? (list (cons 6 #\J) (cons 5 #\D)))
#f
> (deck? (list (cons 6 #\S) (cons 5 #\D)))
#t
>
```

3.7

valueOf function. It sumsup the 2 cards given

```
> (valueOf (list (cons 3 #\S)
                 (cons 5 #\H)))
8
> (valueOf (list (cons #\K #\S)
                 (cons #\Q #\H)))
1.0
>
```

3.8

do-suite function

```
> (do-suite #\D)
((1 . #\D) (2 . #\D) (3 . #\D) (4 . #\D) (5 . #\D) (5 . #\D) (7 . #\D) (#\K . #\D) (#\Q . #\D) (#\J . #\D))
> (do-suite #\C)
((1 . #\C) (2 . #\C) (3 . #\C) (4 . #\C) (5 . #\C) (5 . #\C) (7 . #\C) (#\K . #\C) (#\Q . #\C) (#\J . #\C))
> (do-suite #\S)
((1 . #\S) (2 . #\S) (3 . #\S) (4 . #\S) (5 . #\S) (5 . #\S) (7 . #\S) (#\K . #\S) (#\Q . #\S) (#\J . #\S))
> (do-suite #\H)
((1 . #\H) (2 . #\H) (3 . #\H) (4 . #\H) (5 . #\H) (5 . #\H) (7 . #\H) (#\K . #\H) (#\Q . #\H) (#\J . #\H))
>
```

3.9

deck

```
> deck
((1 . #\H)
 (2 . #\H)
 (3 . #\H)
 (4 . #\H)
 (5 . #\H)
 (6 . #\H)
 (7 . #\H)
 (#\J . #\H)
 (#\Q . #\H)
 (#\K . #\H)
 (1 . #\S)
 (2 . #\S)
 (3 . #\S)
 (4 . #\S)
 (5 . #\S)
 (6 . #\S)
 (7 . #\S)
 (#\J . #\S)
 (#\Q . #\S)
 (#\K . #\S)
 (1 . #\C)
 (2 . #\C)
 (3 . #\C)
 (4 . #\C)
 (5 . #\C)
 (6 . #\C)
 (7 . #\C)
 (#\J . #\C)
 (#\Q . #\C)
 (#\K . #\C)
 (1 . #\D)
 (2 . #\D)
```

3.10

The deck->strings function changes the whole deck into strings

```
> (deck->strings deck)
("1H"
 "2H"
 "3H"
 "4H"
 "5H"
 "6H"
 "7H"
 "JH"
 "QH"
 "KH"
 "1S"
 "2S"
 "3S"
 "4S"
 "5S"
 "6S"
 "7S"
 "JS"
 "QS"
 "KS"
 "1C"
 "2C"
 "3C"
 "4C"
 "5C"
 "6C"
 "7C"
 "JC"
 "QC"
```

3.11

the probability

```
> (probability > 7 (list (cons 7 #\S) (cons 3 #\C) (cons #\K #\D)))
0
> (probability = 7 (list (cons 7 #\S) (cons 3 #\C) (cons #\K #\D)))
1
> (probability < 3 (list (cons 7 #\S) (cons 3 #\C) (cons #\K #\D)))
1
> (probability < 3 (list (cons 7 #\S) (cons 2 #\C) (cons #\K #\D)))
2
>
```

4. TESTING

Provide a test plan covering all of your functions and the results of applying the tests.

Card

In order to test out the cards, we first needed to find out what cards are in a deck of cards and we needed to check out the validity of the card so they are between 1-7 and if it goes above 7.5 the cards will become invalid.

suite and numeral

the suite is the type of cards which are in found in the deck and they are the hearts,diamonds,spade and clubs.What it also does is that it gives you the card which you roll for for example if you roll for the 5 of spades, it will give you the 5 of spades. In addition the number of the card is between 1-7 and if you go above the limit the card will become invalid

face

The face cards are the cards which have an image on them which is the queen,king and the jack and each one of them are worth 0.5 meaning that you need additional cards to make 7.5

5. EVALUATION

*Evaluate your implementation and discuss what you would do if you had more time to work on the code. Critically reflect on the following point and write **300-400 words overall**.*

Points for reflection:

1. What went well with the project was that the cards were correctly identified and the limit of the cards were correctly placed. In addition when adding the numbers it will always end up to be the correct numbers. However some difficulties which have occurred is that coding the probability is much more difficult due to the fact that certain aspects are hard to understand. The value of rounds was difficult because of the complex function.
2. What was learnt that this type of work needs to be done step-by-step because it requires time and patience to understand the concepts. Furthermore the project regarding the development is slow because it requires a deep understanding.
3. If a similar task was to appear the thing you need to is to break it down into different parts. The first thing to do would be to read what is required of us. After doing that we will asign the various tasks to me and my partner to do which will help complete the task.
4. The thing which me and my partner are able to derive from the coursework is that we are able to properly manage our time properly which will help reduce our stress because the time is being managed properly,patience due to the fact that certain aspects require more attention due to it being much harder to complete.

6. GROUP PRO FORMA

Describe the division of work and agree percentage contributions. The pro forma must be signed by all group members and an identical copy provided in each report. If you cannot agree percentage contributions, please indicate so in the notes column and provide your reasoning.
(THIS SECTION SHOULD BE THE SAME FOR BOTH PARTNERS)

Partner ID	Tasks/Features Completed	%Contribution	Signature	Notes
1	001155347	50	MNN	
2	001153094	50	SM	
	Total	0		