CS3723 Final Exam Spring 2020 Section 002 Name \_\_\_Darin Soeung\_\_\_\_\_

©Copyright 2020 by Larry W. Clark, this may not be copied to any website.

**Rename this file using your *lastnameFirstname*.docx**

I certify that I will not share Part1, Part2, and/or the Extras or any answers with other students. I will also not post those onto any website other than submitting this for grading.

Initials: \_\_\_DS\_\_\_ Date: \_\_\_5/14/2020\_\_\_\_ (If this is not initialized, you will receive a 0 on the Final Exam.)

Students who asked me to share my exam or answers (prior to or during) or communicated with me during the exam:

|  |  |  |
| --- | --- | --- |
| Student Name | Prior to the Exam (mark with Y) | During the Exam (mark with Y) |
|  |  |  |
|  |  |  |
|  |  |  |

Students who I asked to share their exam or answers with me prior to or during the exam:

|  |
| --- |
| Student Name |
|  |
|  |
|  |
|  |
|  |

Initials: \_\_\_ DS \_\_\_ Date: \_\_\_\_5/14/2020\_\_\_\_\_

Part 2 Scores:

|  |  |  |
| --- | --- | --- |
| 1 | 50 |  |
| 2 | 50 |  |
| 3 | 10 |  |
| Total | 110 | 0.0 |
| 300pt Total | 330 | 0.0 |

(50pts) 1. Show code for two **recursive** LISP functions (see the examples):

(**findInPair** *atm* *pairList*) which looks for *atm* in the *pairList.* If it matches the first value of a pair, it returns the corresponding second value of the pair. If not found in any of the pairs, **findInPair** returns *atm*.

(**replacePair** *list pairList*) which is passed a top-level *list* and a list containing multiple pairs. It returns a list that has replaced atoms in *list* that match the first item of a pair with the second item in the pair.

**Notes:**

1. **You may only use LISP features/functions we discussed in course notes.**

2. **replacePair** can use **findInPair**.

3. Hint for **replacePair**: you are constructing a new list.

> (findInPair 'X '( (S V) (I O) (P E)))

X

> (findInPair 'P '( (S V) (I O) (P E)))

E

> (replacePair '(L I S P) '( (S V) (I O) (P E)))

(L O V E)

(defun findInPair (atm pairList)

(cond

((NULL pairList) atm)

((memset atm (car pairList)) (cadar pairList) )

(T (findInPair atm (cdr pairList)))

)

)

(defun replacePair (list pairList)

(cond

((NULL pairList)())

((memset (car list) (car pairList))

(cons(findInPair (car list) (car pairList)) list)))

(T (replacePair (cdar list) pairList))

)

)

(50pts) 2. You must write a **Python program** which determines the winner when playing blackjack. It is passed a filename. Sample Input File for one game:

DEALER HA H8

PLAYER H3 D7 S10

PLAYERD8 CK

PLAYER CQ S9

PLAYER C7 S5 SJ

Meaning of that data:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Data** | **Who** | **Cards** | **Points** | **Winner?** |
| DEALER HA H8 | Dealer | ace of hearts, 8 of hearts | 11+8=19 |  |
| PLAYER H3 D7 S10 | Player 1 | 3 of hearts, 7 of diamonds,10 of spades | 3+7+10=20 | player |
| PLAYERD8 CK | Player 2 | 8 of diamonds, king of clubs | 8+10=18 | dealer |
| PLAYER CQ S9 | Player 3 | queen of clubs, 9 of spades | 10+9=19 | dealer |
| PLAYER C7 S5 SJ | Player 4 | 7 of clubs, 5 of spades, jack of spades | 7+5+10=22 | dealer |

**Rules:**

The players are not playing against each other. Each is simply competing against the dealer.

* If a player exceeds 21 total points, he loses.
* If a player has more points than the dealer, but doesn't exceed 21, player wins.
* Ties go to the dealer.
* Point values:
  + Each numeric card has its point value (e.g., H3 is 3 points)
  + Each face card (jack, queen, king) is worth 10 points
  + Aces are worth 11 points. You will receive a bonus of 2 pts if you properly handle having them worth either 1 or 11 points by using 11 only if not going over a total of 21. Otherwise use 1 point. This includes handling multiple aces.

**Notes:**

1. Your program must call the function calculateScore(*hand*), *hand* represents the cards that were dealt for one person's hand (e.g., "HA H8", "HE D7 S10"). It determines the score based on the rules above.
2. You must show your code for calculateScore and your program.
3. You may assume that you receive the dealer's hand before any of the players.
4. **Your program:**
   * Must validate the command argument
   * Must call your calculateScore function (multiple times) to determine the score for the dealer and each player.
   * Output the score of the dealer, the score of each player, and specify who won as shown below
5. **Your code must only use Python features we discussed in my course notes.**
6. **You may receive less points for a less efficient approach.**

**Sample Output:**

DEALER HA H8 19

PLAYER H3 D7 S10 20 PLAYER

PLAYER D8 CK 18 DEALER

PLAYER CQ S9 19 DEALER

PLAYER C7 S5 SJ 22 DEALER

Import sys

cardDict = {“S1”:1, “S2”:2, … “DK”:10, “HK”:10, “CK”:10, “SK”:10, “DQ”:10, “HQ”:10, “CQ”:10, “SQ”:10, “DJ”:10, “HJ”:10, “CJ”:10, “SJ”:10, “DA”:11, “HA”:11, “CA”:11, “SA”:11 }

Def calculateScore(hand)

HandM = hand.split()

score = 0

bstring = “”.join(handm)

For cards in handM:

handVal = cardDict.get(cards, “NF”)

if handVal == “NF”:

print(“invalid card: ”, cards)

continue

score += handVal

print (bstring + “ ” + str(score))

return score

)

#main program

If len(sys.argv) < 2:

Print(“Please insert a file for hands”);

Sys.exit(1)

File = open(sys.argvp[1], “r”)

While True:

Line = file.readline()

If line == “”:

Break

Line = line.rstrip()

If “DEALER” in Line:

DealerScore = calculateScore(Line)

Else:

Score = calculateScore(line)

Print(“\n”)

If Score > DealerScore:

Print(“ ” + str(score) + “ PLAYER”);

Else:

Print(“ ” + str(score) + “ DEALER”);

(10pts) 3. Show LISP code for the macro (**CountIn** *matchValue arg1 arg2 …*)which is passed a match value and a variable number of additional *arguments*. It must count the number of occurrences of the match value in the evaluated value of the other arguments.

**You may only use LISP features/functions we discussed in course notes.**

> (setf X 1)

> (**CountIn** (+ x x) 5 2 (\* x 2) (- 3 x) (\* x 3))

**3**

**Values:** 2 5 2 2 2 3

To refresh your mind about macros, here is the code from the notes on the switch macro:

(defmacro switch (value &rest pairs)

(let ((g (gensym)) )

`( do ( (,g ' ,pairs (cdr ,g)) )

( (null ,g) NIL )

(if (or

(eql 'DEFAULT (caar ,g))

(eql ,value (eval (caar ,g)))

)

(return (eval (cadar ,g)))

) ) ) )