**BAHRIA UNIVERSITY, ISLAMABAD CAMPUS**

DEPARTMENT OF COMPUTER SCIENCE

CLASS/SECTION: BSAI-3A

(Spring 2023 Semester)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

GROUP PARITICIPANTS:

|  |  |
| --- | --- |
| **NAME** | **ENROLLMENT** |
| FARHAN AHMAD | 01-136221-052 |
| TAIFOOR ASRAR | 01-136221-019 |
| TAHA HASNAT | 01-136221-018 |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**COURSE:**

ARTIFICIAL INTEELLIGENCE

**SUBMITTED TO:**

SIR ADIL KHAN

**DATE OF SUBMISSION:** 5/6/2023

**CODE:**

;; Poker Game in Lisp

;; Card data structure

(defstruct card

rank

suit)

;; Initialize the players' hands

(defvar \*player-hand\* nil)

(defvar \*computer-hand\* nil)

;; Function to create a new deck of cards

(defun create-deck ()

(loop for rank from 2 to 14

nconc (loop for suit in '(:hearts :diamonds :clubs :spades)

collect (make-card :rank rank :suit suit))))

;; Function to shuffle the deck

(defun shuffle-deck (deck)

(loop for i from (1- (length deck)) downto 1

for j = (random (1+ i))

collect (let ((temp-card (nth i deck)))

(setf (nth i deck) (nth j deck))

(setf (nth j deck) temp-card))))

;; Function to deal the initial hands

(defun deal-hands (deck)

(let ((player-card (pop deck))

(computer-card (pop deck)))

(setf \*player-hand\* (list player-card))

(setf \*computer-hand\* (list computer-card))

(push (pop deck) \*player-hand\*)

(push (pop deck) \*computer-hand\*)))

;; Function to evaluate the rank of a hand

(defun evaluate-hand (hand)

;; Placeholder implementation

;; Return a random rank for demonstration purposes

(random 10))

;; Function to determine the winner

(defun determine-winner ()

(let ((player-rank (evaluate-hand \*player-hand\*))

(computer-rank (evaluate-hand \*computer-hand\*)))

(format t "Player hand: ~{~A of ~A~^, ~} (Rank: ~A)~%"

(mapcar #'(lambda (card) (list (card-rank card) (card-suit card))) \*player-hand\*)

player-rank)

(format t "Computer hand: ~{~A of ~A~^, ~} (Rank: ~A)~%"

(mapcar #'(lambda (card) (list (card-rank card) (card-suit card))) \*computer-hand\*)

computer-rank)

(cond

((> player-rank computer-rank) "Player wins!")

((< player-rank computer-rank) "Computer wins!")

(t "It's a tie!"))))

;; Function to play a round of poker

(defun play-poker ()

(let\* ((deck (shuffle-deck (create-deck))))

(deal-hands deck)

(format t "=== Poker Game ===~%")

(format t "Player hand: ~{~A of ~A~^, ~}~%"

(mapcar #'(lambda (card) (list (card-rank card) (card-suit card))) \*player-hand\*))

(format t "Computer hand: ~{~A of ~A~^, ~}~%"

(mapcar #'(lambda (card) (list (card-rank card) (card-suit card))) \*computer-hand\*))

(format t "=================~%")

(format t "Determining the winner...~%")

(format t "=================~%")

(format t "~A~%" (determine-winner))))

;; Start the game

(play-poker)