**Project - 3**

****

**Overview**

The project is to create a card-based role-playing game between two (computer) players in which they battle each other with action cards.

***Game Play***

In each turn, there is a battle in which the outcome is determined by the card types and values. After each turn, the current cards are discarded into the abyss. In some outcomes, one player may gain one card, or one player may lose one card, or both may lose their cards into the abyss. The game continues until one player has lost all their cards. The player with any remaining cards wins.

***Cards***

Each player has a deck of cards, implemented with a linked list. Each card is a node in a linked list. There are three types of cards: ATTACK, DEFEND, and RUN. Here are the scenarios for each battle showdown between the two players.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **ATTACK** | **DEFEND** | **RUN** |
| **ATTACK** | If both players ATTACK, the player with the higher value wins. The winning player gets a new card, which will be added to their card list in the descending order of the card value. The losing player will lose their next card into the abyss. In a tie, nothing happens. | If one player ATTACKs and the other player DEFENDs, the player with the higher value wins. If the ATTACKing player wins, they get a new card added to their card list in the descending order of the card value and nothing happens to the DEFENDing player. If the DEFENDing player wins or ties, then the ATTACKing player loses their next card into the abyss. | If one player ATTACKs and the other player RUNs, the RUNning player loses their next card into the abyss. |
| **DEFEND** |  | If both players DEFEND, nothing happens. | If one player DEFENDs and the other player RUNs, the DEFENDING player gets a new card added to their card list in the descending order of the card value. The RUNning player loses their next card into the abyss. |
| **RUN** |  |  | If both players RUN, they both lose their next cards into the abyss. |

**Task 1 - cards.h**

* Declare a **struct** that will represent a **Card**, which will also be a node in a linked list. It will contain a **CardType** (see **enum** below) and an integer as the value.
* Declare all prototypes for the required functions (see Task 2) as well as the Header Guard.
* Include this **enum** for the **CardType**:

**typedef enum CardType { ATTACK, DEFEND, RUN } CardType;**

**Task 2 - cards.c**

Write the following functions which are typical linked list functions adapted for the game play:

* **Card\* createCard()** - This function dynamically allocates a new **Card struct**object and returns a pointer to that struct object which will later be used to insert into a linked list. There are three types of cards ATTACK, DEFEND, and RUN. ATTACK and DEFEND cards also have a value. You will assign a card type based on these random chances:
  + 40% - ATTACK: the value is a random number between 1 and 5 inclusive.
  + 50% - DEFEND: the value is a random number between 3 and 8 inclusive.
  + 10% - RUN: the value is a random number between 1 and 8 inclusive. The value of a RUN card is only used for sorting purposes.
* **Card\* removeCard( Card \*head )** - This function removes and deallocates the first node in the linked list that head is pointing at. It returns the new head of the linked list.
* **Card\* addCard( Card \*head, Card \*c )** - This function adds a new **Card struct** object to the linked list that head is pointing at. It is assumed that a new **Card struct**object is being passed into this function as parameter **c**. This function will add the new node in **descending** order of its value regardless of the action.
* **int getLength( Card \*head )** - This function returns the length of the linked list that head is pointing at.
* **void printCard( Card \*head )** - This function prints a single card with abbreviations: A for ATTACK, D for DEFEND, and R for RUN and their corresponding values. Example: A5 means an ATTACK worth 5 points.
* **void printCards( Card \*head )** - This function traverses the linked list that head is pointing at. It will call printCard(). It should print out from front to back. Example: R6 D5 D4 A4 D3
* **Card\* buildCards( int n )** - This function builds a stack of cards using a linked list. The parameter n determines how many cards are created. It will use **createCard()** and **addCard()**. It returns the head of the new linked list.
* **Card\* destroyCards( Card \*head )** - This function is the destructor for a linked list that head is pointing at. It should return **NULL**.

**Task 3 - project3-main.c**

Write the **main()** to do the following:

* The program will accept one additional command line argument, which is the number of cards players will start with. Your program must determine if there is this command line argument:
  + If the command line argument does not exist, print an error, and end the program.
  + If the command line argument, check to see if it is a number greater than 0. If it is not, print an error and end the program. Use the C function**atoi()** to help convert a string to an int.
* If you have number greater than zero for that additional command line argument, you can start the game:
  + Build each player’s Card linked list based on the size passed in through the command line arguments. Print out each player's cards using **printCards()**.
  + Start the game loop. The game continues so long as both players have cards to play with. In each turn (iteration of the loop):
    - Print out the round number starting at 1. Print the player name and the number of cards currently remaining in brackets. Also print each player’s first card that is being played currently, along with its value. For example, Player 1 (12): D7 & Player-2 (12): R8.
    - Implement the outcomes for each permutations of player actions. There are 9 permutations of ATTACK, DEFEND, and RUN (AA, AD, AR, DA, DD, DR, RA, RD, RR). You can have an if-statement for each of these permutations. Print out descriptive information about what is happening, who wins, and assess the reward/punishment on both players as determined by the rules above. This involves creating/adding a new card to a player’s linked list if they win and/or removing another card from a player’s linked list. There are also cases when there are no rewards nor punishment. You will use your functions to assess these outcomes.
    - At the end of each turn, remove the front of the linked list from each player. This discards the cards that the players used in this round. Remember to do remove this current card before applying the reward/punishment for each of the above cases. i.e. the current card should be discarded first before removing the next card as penalty or adding a new card as reward.
  + Determine the winner and print out the outcomes. There is also the case in which both players lose.

**Task 4: Create a makefile**

Create a makefile to compile and link. The grader will compile your code using your makefile.

**Submission**

Be sure that your code follows the class coding style requirements. Your output should be similar in format as compared to the sample output (shown below as well as attached in a separate txt file). Select all your program files and create a zip file from that. Name this zip file as **abc123.zip** and submit on Blackboard.

**Sample Output**

See below or the attached **project3-output.txt** file for some sample output. Your output should be similar.

**Rubric**

See attached **project3-rubric.txt** for the grading rubric.

**SAMPLE – OUTPUT – RUN-1**

./project3 12

============= PLAYER 1 V PLAYER 2 SHOWDOWN ============

Start size: 12 cards

Player 1 starting cards: D7 D6 R6 R6 D5 D4 A4 D3 D3 D3 D3 D3

Player 2 starting cards: R8 R7 R7 R6 A5 A5 A5 A4 A4 D3 D3 A1

----- ROUND 1 -----

Player 1 (12): D7

Player 2 (12): R8

Player 1 DEFENDs and Player 2 RUNs.

Player 1 gets a new card. Player 2 loses their next card into the abyss.

----- ROUND 2 -----

Player 1 (12): D6

Player 2 (10): R7

Player 1 DEFENDs and Player 2 RUNs.

Player 1 gets a new card. Player 2 loses their next card into the abyss.

----- ROUND 3 -----

Player 1 (12): R6

Player 2 (8): A5

Player 2 ATTACKs and Player 1 RUNs.

Player 1 loses their next card into the abyss.

----- ROUND 4 -----

Player 1 (10): D5

Player 2 (7): A5

Player 2 ATTACKs and Player 1 DEFENDs.

Player 2 loses and Player 1 survives.

Player 2 loses their next card into the abyss.

----- ROUND 5 -----

Player 1 (9): D4

Player 2 (5): A4

Player 2 ATTACKs and Player 1 DEFENDs.

Player 2 loses and Player 1 survives.

Player 2 loses their next card into the abyss.

----- ROUND 6 -----

Player 1 (8): A4

Player 2 (3): D3

Player 1 ATTACKs and Player 2 DEFENDs.

Player 1 wins. Player 1 gets a new card.

----- ROUND 7 -----

Player 1 (8): D5

Player 2 (2): D3

Both players DEFEND.

Nothing happens.

----- ROUND 8 -----

Player 1 (7): A4

Player 2 (1): A1

Both players ATTACK.

Player 1 wins and gets a new card. Player 2 loses their next card into the abyss.

============ GAME OVER =============

Player 1 ending cards: D3 D3 D3 D3 D3 D3 A2

Player 2 ending cards:

Player 2 ran out of cards. Player 1 wins.

The end.

**SAMPLE – OUTPUT – RUN-2**

./project3 12

============= PLAYER 1 V PLAYER 2 SHOWDOWN ============

Start size: 12 cards

Player 1 starting cards: R8 R8 D7 D6 A5 A5 A3 A3 A2 R2 R2 A2

Player 2 starting cards: D6 R6 D4 D3 D3 A3 D3 D3 R2 A2 A2 A1

----- ROUND 1 -----

Player 1 (12): R8

Player 2 (12): D6

Player 2 DEFENDs and Player 1 RUNs.

Player 2 gets a new card. Player 1 loses their next card into the abyss.

----- ROUND 2 -----

Player 1 (10): D7

Player 2 (12): R6

Player 1 DEFENDs and Player 2 RUNs.

Player 1 gets a new card. Player 2 loses their next card into the abyss.

----- ROUND 3 -----

Player 1 (10): R7

Player 2 (10): A4

Player 2 ATTACKs and Player 1 RUNs.

Player 1 loses their next card into the abyss.

----- ROUND 4 -----

Player 1 (8): A5

Player 2 (9): D3

Player 1 ATTACKs and Player 2 DEFENDs.

Player 1 wins. Player 1 gets a new card.

----- ROUND 5 -----

Player 1 (8): A5

Player 2 (8): D3

Player 1 ATTACKs and Player 2 DEFENDs.

Player 1 wins. Player 1 gets a new card.

----- ROUND 6 -----

Player 1 (8): R7

Player 2 (7): A3

Player 2 ATTACKs and Player 1 RUNs.

Player 1 loses their next card into the abyss.

----- ROUND 7 -----

Player 1 (6): A3

Player 2 (6): D3

Player 1 ATTACKs and Player 2 DEFENDs.

Player 1 loses and Player 2 survives.

Player 1 loses their next card into the abyss.

----- ROUND 8 -----

Player 1 (4): R2

Player 2 (5): D3

Player 2 DEFENDs and Player 1 RUNs.

Player 2 gets a new card. Player 1 loses their next card into the abyss.

----- ROUND 9 -----

Player 1 (2): A2

Player 2 (5): D6

Player 1 ATTACKs and Player 2 DEFENDs.

Player 1 loses and Player 2 survives.

Player 1 loses their next card into the abyss.

============ GAME OVER =============

Player 1 ending cards:

Player 2 ending cards: R2 A2 A2 A1

Player 1 ran out of cards. Player 2 wins.

The end.