Din Bostandzic
CSE 13S

Assignment 1 - Left, Right, Center Program Design

The program is an implementation of a left, right, center, game which rules are as follows: the user inputs a random seed along with total players (at least 2, max 10), then each player gets $3, which corresponds to number of die rolls (ex. $3 is 3 rolls … $0 is pass) starting from first player, the rolls can be either L, R, C, P, meaning that if player rolls L, gives $1 to player to left, R is for giving player to right $1, C is for adding $1 to the pot, while P passes the turn, then the next player (right of current) goes. The program runs infinitely until the condition is met of only one player with cash remaining is the winner. To approach it the following functional decomposition has been implemented, along with the supporting Pseudocode:

3.0 left right center
- 3.1 left(in pos as unsigned 32 bit integer, in players as unsigned 32 bit integer)
- 3.2 right(in pos as unsigned 32 bit integer, in players as unsigned 32 bit integer)
- 3.3 rand_num(out rand_val as integer)
- 3.4 play_game(in money_players as array of integer, in names as array string, in die as user-defined type faces, in player_pos as integer, in player as integer)
- 3.5 die_rules(in play as integer, in times as integer, in names as array of string, in die as user-defined type faces, in money_players as array of integer, in player as integer)
- 3.6 player_in_checker(in playercash as integer)
- 3.7 die_choice(in die_roll as integer)

Data Design

Define POT as static integer initialized to 0
Define PLAYERSIN as static integer initialized to 0
Declare faces enumerated list of constants with LEFT, RIGHT, CENTER, PASS of type faceim

Declare array die of type faces initialized to LEFT, RIGHT, CENTER, PASS, PASS, PASS
Declare array names of string type initialized to “Happy”, “Sleepy”, “Sneezy”, “Dopey”, “Bashful”, “Grumpy”, “Doc”, “Mirror”, “Snow White”, “Wicked Queen”
Declare player and seed, play_pos as integers initialized to 0
Declare money_players as integer array of size player

Main Module Design

Begin Main
    Display “Random Seed: ”
    Input seed
    Display “How many players? ”
    Input player
    Begin While
    While player is less than or equal to 1 or player is greater than 10

```
            Display “Enter valid number of players, must be at least 2 players, max 10”
            Input player
        End While
        Assign player value to PLAYERSIN
        Begin For
        For integer i declared and initialized to 0, i less than player, increment i
            money_players[i] = 3
        End For
        Call play_game module pass in money_players as array of integers, names as array of
                string type, die as array of faces type, play_pos as integer, player as integer
End Main
```

rand_num Data Design

```
Declare rand_val as integer
```

rand_num Module Design

```
Begin rand_num
        Assign value of  (rand() modulo 6) to rand_val
        Return value of rand_val
End rand_num
```

play_game Data Design

```
Declare record_spot as integer and initialize to 0
```

play_game Module Design

```
Begin play_game
        Begin While
        While 1
            Begin If
            If money_players[play_pos] >= 3
                Call die_rules module pass in play_pos as int, 3, names as array of string,
              die as array of face type, money_players as array of integer, player as integer

                Assign value of called right module pass in play_pos as integer, player as
                 integer, to play_pos
            End If
            Begin Else If
            Else if money_players[play_pos] == 2
```

```
                Call die_rules module pass in play_pos as int, 2, names as array of string,
            die as array of face type, money_players as array of integer, player as integer

                Assign value of called right module pass in play_pos as integer, player as
                  integer, to play_pos
        End Else If
        Begin Else If
        Else if money_players[play_pos] == 1
                Increment PLAYERSIN by 1

                Call die_rules module pass in play_pos as int, 1, names as array of string,
            die as array of face type, money_players as array of integer, player as integer

                Assign value of called right module pass in play_pos as integer, player as
                  integer, to play_pos
        End Else If
        Begin Else if
        Else if  money_players[play_pos] == 0
                Decrement PLAYERSIN by 1
                Assign value of called right module pass in play_pos as integer, player as
                integer, to play_pos
        End Else If
        Begin For
        For integer i declared and initialized to 0, i less than player, increment i
                Begin If
                If  money_players[i] > 0
                        Assign to record_spot value of i
                End If
        End For
        Begin If
        If  PLAYERSIN == 0
                Display  names[record_spot] “wins the $” POT “ with ”
                money_players[record_spot] “ left in the bank!"
                Break out of while loop
        End If
    End While
End play_game
```

<u>die_rules Data Design</u>

```
Declare check_string as string initialized to “left”
Declare check_string2 as string initialized to “right”
```

Declare check_string3 as string initialized to “center”

<u>die_rules Module Design</u>

Begin die_rules

Display names[play] “rolls.. ”

Begin For

For integer i declared and initialized to 0, i less than times, increment i

Declare dice_num as integer initialized to value returned by call rand_num module

Declare die_value as integer initialized to value of die array indexed by dice_num

Declare die_play as string initialized to value returned by call die_choice module pass in die_value

Begin If

If die_play == check_string

Declare left_pos as integer initialized to value returned by call left module, pass in play as integer, player as integer

money_players[left_pos] += 1;

money_players[play] -= 1;

Call player_in_checker module pass in value of money_players[left_pos]

Display names[left_pos] “gives $1 to ”

End If

Begin Else If

Else if die_play == check_string2

Declare right_pos as integer initialized to value returned by call right module, pass in play as integer, player as integer

money_players[right_pos] += 1

money_players[play] -= 1

Call player_in_checker module pass in value of money_players[right_pos]

Display names[right_pos] “gives $1 to ”

End Else If

Begin Else If

Else if die_play == check_string3

Increment POT by 1

money_players[play] -= 1

Begin If

If money_players[play] == 0

Decrement PLAYERSIN by 1

End If

Display “puts $1 in the pot ”

End Else if

Begin Else

Else

Display die_play “gets a ”

End Else
End For
End die_rules

die_choice Module Design

Begin die_choice
Begin switch statement of die_roll value
Case of 0
Return “left”
Break out of case
Case of 1
Return “right”
Break out of case
Case of 2
Return “center”
Break out of case
Default case
Return “pass”
Break out of case
End switch statement
End die_choice

player_in_checker Module Design

Begin player_in_checker
Begin If
If playercash == 1
Increment PLAYERSIN by 1
End If
End player_in_checker