COM2001: Advanced Programming Topics

Assignment 2

Design

Data Types

In this assignment, four new data types are introduced.

1. DomsPlayer: Represent a player function

```
type DomsPlayer = Hand -> Board -> (Domino, End)
```

2. Turn: An algebraic data type to represent a turn for Player 1 or Player 2

```
data Turn = P1 | P2 deriving (Eq,Show)
```

3. Hands: Use a tuple to represent a pair of hands, i.e. Player 1's hand and Player 2's hand

```
type Hands = (Hand, Hand)
```

4. Scores: Use a tuple for Player 1's score and Player 2's score

```
type Scores = (Int, Int)
```

Variable names used in this document**:**

dom	**:**	A domino
brd	**:**	A board
end, L, R	**:**	An end, Left or Right
hand	:	A hand of a player
hands	**:**	A pair of hands
p1, p2	**:**	Player 1 and Player 2
score	**:**	The score after playing a domino
seed	**:**	An integer used to initialise the random number generator

Functions

1. simplePlayer: It will be a recursive function, which calls itself until the first domino in its hand can be played

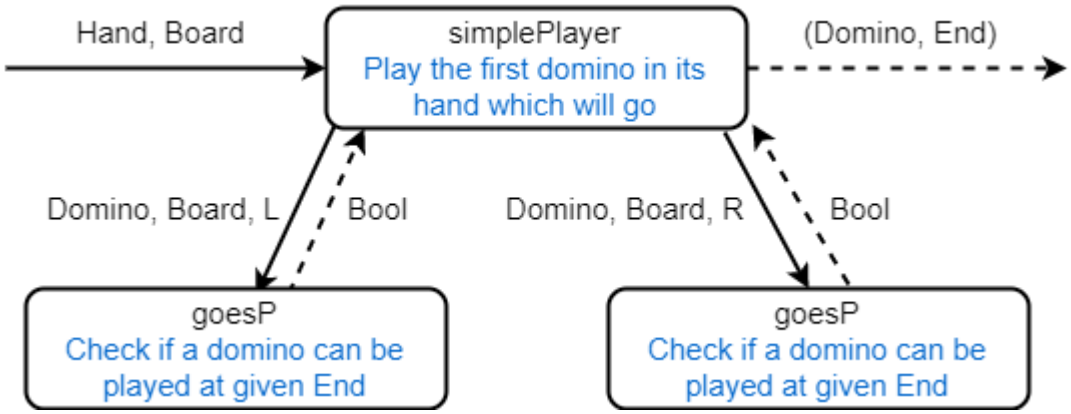

Figure 1. Design of simplePlayer

Steps:
1. Call **simplePlayer (dom:rhand) brd**
2. If the first **dom** can be played at **L**, then return **(dom, L)**
3. Else if the first **dom** can be played at **R**, then return **(dom, R)**
4. Otherwise call **simplePlayer rhand brd**

2. hsdPlayer: It will use helper functions to find the maximum scoring domino played at **L**, and the maximum scoring domino played at **R**, then compare and return the maximum between the two dominoes

2.1. maxHsd

Steps:
1. Compare the dominoes returned by **hsdLeft** and **hsdRight**
2. Return the maximum domino as **(score, dom, end**)

2.2. hsdLeft

Steps:
1. Play all dominoes that can be played at **L**
2. Use **scoreboard** function to calculate the score of each domino
3. Use **maximum** function to find the maximum domino
4. Return the maximum domino as **(score, dom, L)**

2.3. hsdRight

Steps:
1. Play all dominoes that can be played at **R**
2. Use **scoreboard** function to calculate the score of each domino
3. Use **maximum** function to find the maximum domino
4. Return the maximum domino as **(score, dom, R)**

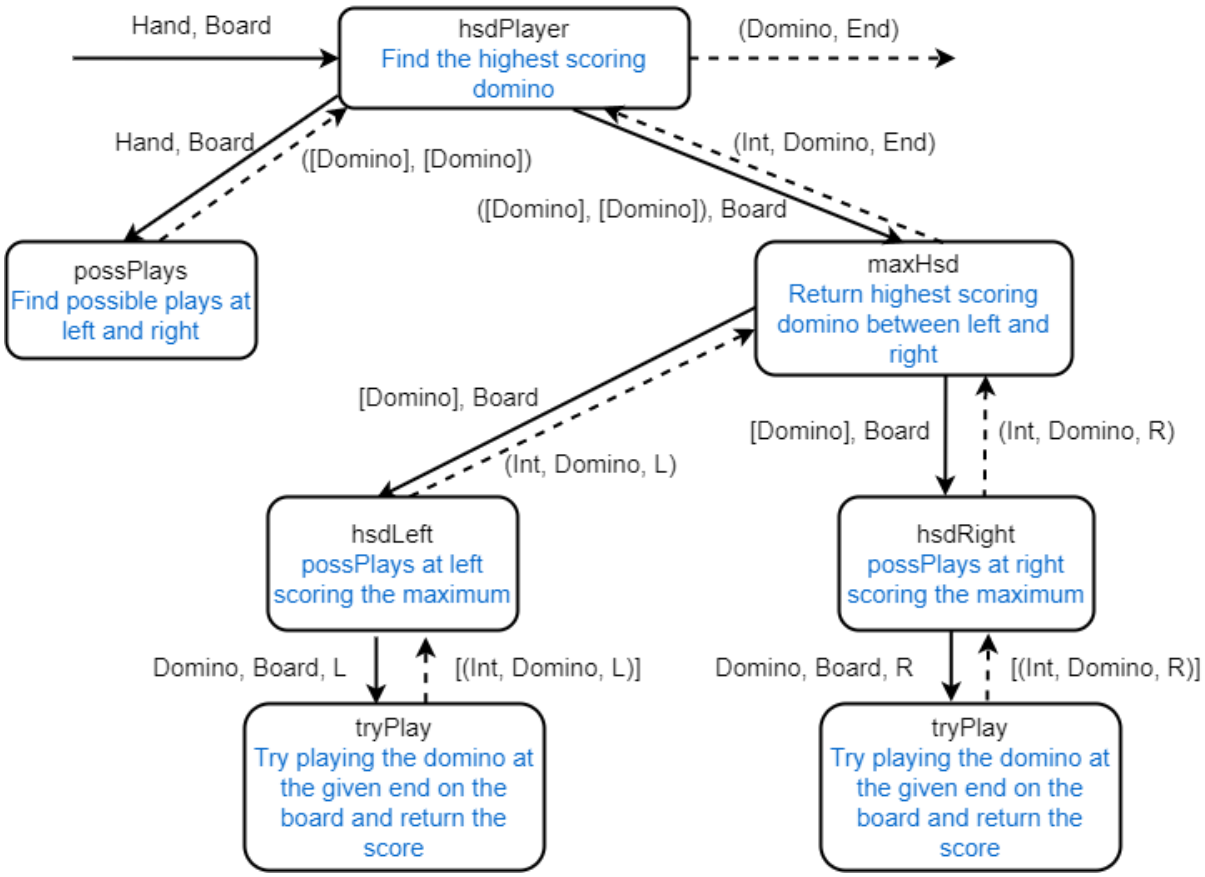

Figure 2. Design of hsdPlayer

Steps:

1. Call **hsdPlayer hand brd**
2. **hsdPlayer** will call **possPlays hand brd** to return the possible plays for **L** and **R**
3. Call **maxHsd hands brd,** which the **hands** are the tuple of list of dominoes returned by **possPlays**
4. **maxHsd** will call **hsdLeft** and **hsdRight**, and use **max** function to find the maximum domino
5. **hsdLeft** and **hsdRight** will return the maximum domino played at **L** and **R** respectively, as mentioned in **2.2** and **2.3** above

3. shuffleDoms: Provide a seed, and it will generate a full list of dominoes in random order

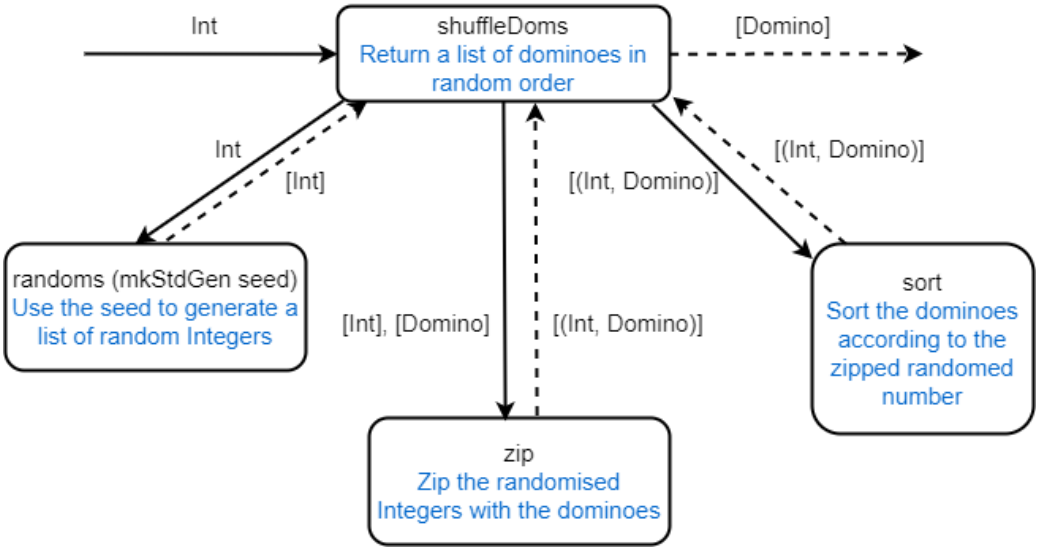

Figure 3. Design of shuffleDoms

Steps:
1. Call **shuffleDoms seed**
2. Use **randoms** and **mkStdGen** to generate a list of random integers
3. **Zip** the list of integers with the full dominoes set
4. **Sort** the list, and then return the list with the generated number removed

4. playDomsRound: Given two players (**simplePlayer** or **hsdPlayer**), return the final score of each player in pair

4.1. createHands: Simply take the first 9 dominoes as **Player 1's** hand, and the next 9 dominoes as **Player 2**'s hand

4.2. startTurn

Steps:
1. If **endGame** returns true, then return the final score
2. Else if **p1CanPlay** returns true, then call **p1Turn**
3. Else if **p2CanPlay** returns true, then call **p2Turn**
4. Call **startTurn** again until **endGame** returns true

4.3. endGame: If **Player 1** and **Player 2** are **knocking**, then returns true.

4.4. p1CanPlay: Return **True** if it is **Player 1's** turn and **Player 1** is **not knocking**, or **Player 2** is **knocking**

4.5. p2CanPlay: Return **True** if it is **Player 2's** turn and **Player 2** is **not knocking**, or **Player 1** is **knocking**

4.6. playerPlay: Make a single move, returning the resulting hand and board, and the score for that move

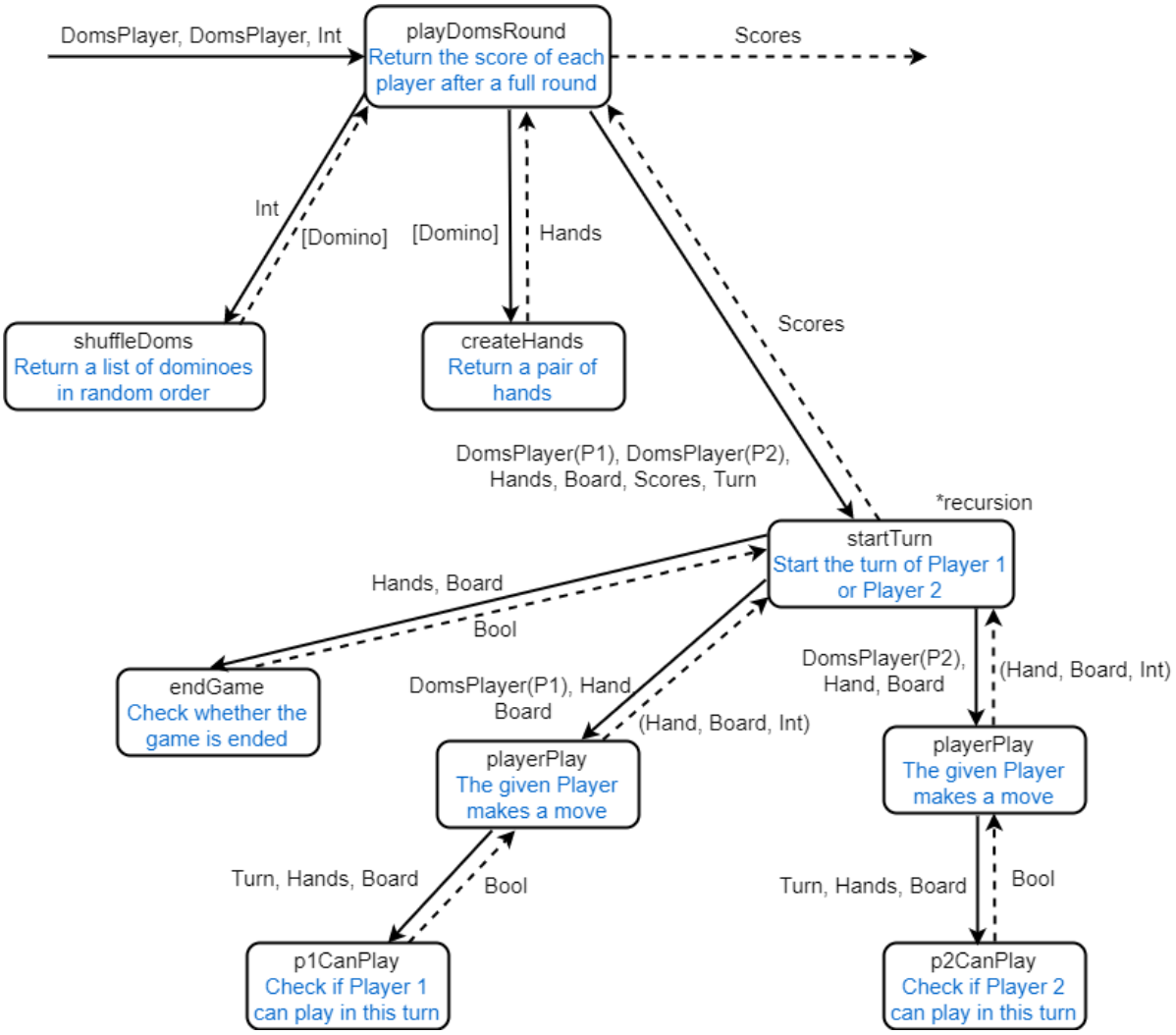

Figure 4. Design of playDomsRound

Steps:

1. Call **playDomsRound p1 p2 seed**
2. **shuffleDoms** will be called, then **createHands** will be called next.
3. After that, **startTurn** will be called, given **`[]`, `(0,0)`, P1** as starting value for the **Board, Scores, Player**
4. **startTurn** will be called recursively until **endGame** returns **True,** as mentioned in **4.2** above. Then it will return the final score back to **playDomsRound**