KHADIJA BUTT BESE 5A CMS ID 111661

ASSIGNMENT # 1: SNAKES & LADDERS

Introduction

Snakes and Ladders is a grid based game which contains 0-100 numbers and some snakes and ladders at some positions on the grid. Player moves forward by rolling a dice and incrementing its position by the random number of dice roll. If the player encounters a snake on its way, it ends up on snake's tail position and if it encounters a ladder, it ends up on the ladder's top. The first player to reach the score of 100 wins.

Approach:

The main idea is that the locations of all snakes and ladders are stored in hashmaps and before updating a user's position, it is checked whether that position contains a snake/ladder or not.

To implement this game following 3 main functions are used:

1. *Dice_roll()*

   This function returns a random number from 1-6 which is used as an increment value for updating user's position. This function is called in user's turn by a button press and automatically on computer's turn.

```
int dice_roll(){
    Random dice_count = new Random();          //generates a random number as dice is rolled
    return dice_count.nextInt(6);
}
```

2. *Snake_ladder(int position)*

   This functions checks after every turn whether the player encounters a snake or ladder or not. The start and ends of all snakes and ladders are stored in a HashMap. It compares the given position argument to HashMap's value. If there is a snake or ladder at that position, it returns the new position or else it returns the same input value hence the update_postion() function simply has to update the current user's position by adding the return value of this function.

```
/*before player moves on a new position, checks whether there
is a snake/ladder there or not and returns new position*/
int snake_ladder(int position){
    int new_position = 0;
    if(snakes.containsKey(position)) {              //checks if player position coincides with snake location
        new_position = Integer.parseInt(snakes.get(position).toString());       //get snake's tail position
        return new_position;                //returns new position
    }
    else
        return position;        //if no snake coincides, new position doesn't change
}
```

3. *Update_position(int current, int dice_count)*

This function simply adds the return value of snake_ladder() function to current player's position and moves user's image to that position. The position is updated by taking a reference to image view from xml file and setting its new coordinates using image.setX() function.

```
void update_position(int current, int dice_value){
        current = current + snake_ladder(current+dice_value);     //snake_ladder function will return increment value
        user.setX( current);
}
```

All the above functions are called inside a while loop which runs unless any of the player score 100 and wins. A Boolean variable determines if its user's turn or computer's. If user's turn, the dice roll button is enables and the user clicks it to generate a random number, else the dice_roll() is called automatically.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    while (max_score<100){
        if( turn == false){         //if its user's turn
            btn.setEnabled(true);           //enable the dice roll button
            dice_count=dice_roll();
            btn.setEnabled(false);
            update_position(current_pos_user, current_pos_user+dice_count);     //update user's position
        }
        else {
            dice_count = dice_roll();           //else if its computer's turn
            update_position(current_pos_comp, current_pos_user+dice_count);         //update comp's position
        }

    }
```