Group Project

1 Monopoly

1.1 Brief description of the problem

We wish to determine which squares a player is most likely to land on during a game of Monopoly. To simplify things, we assume there is only a single player, we ignore everything to do with money, and we also ignore the 'Get out of Jail Free Cards'. The algorithm we will use is:

1. Begin the game on GO;
2. `current` := `current` + dice roll
3. Make a note of the new position.
   - If we land on 'Chance' or 'Community Chest', draw a card;
   - If we land on 'Go To Jail', move to Jail;
   - If we move, make a note of the new position;
4. Go back to step 2

After rolling the dice 100,000 times or so, stop. Familiarise yourself with the standard Monopoly board (see Table 1 in the Appendix of this handout, or search online for more detailed information).

1.2 Dice rolling

When we roll a single die, the *probability mass function* (see MAS1604) is:

$$\Pr[X = k] = 1/6 \quad \text{for } k = 1, \ldots, 6;$$

i.e. all sides of the die are equally likely. This means we can use the **sample** function to simulate a die roll:

```
sample(c(1, 2, 3, 4, 5, 6), 1)
#Or just the same:
sample(seq(1, 6), 1)
#Or:
sample(1:6, 1)
```

To roll two dice, we simply call the function in Listing 1:

Listing 1: Rolling two dice

```
RollTwoDice = function() {
  total = sample(1:6, 1) + sample(1:6, 1)
  return(total)
}
```

Helpful exercises:

- *Call the function* `RollTwoDice` *a few times, and see what you get.*
- *What are the range of possible values* `RollTwoDice` *could return?*
- *If we tried to simulate rolling two dice using* `sample(2:12, 1)`*, why would this be wrong?*

1.3 The Monopoly board

In Monopoly there are forty squares; see Table 1 in the Appendix for a complete list. If we set the first square 'Go' to be number 1, then we can represent all forty squares as a vector in **R**. For example:

```
#This creates a vector of 40 values; all values are initially zero
landings = numeric(40)
```

Then, when we land on a square we simply increase the associated `landings` entry by one. Suppose we landed on 'Old Kent Rd'. Since this is square 2 (see Table 1), we would represent this as:

```
landings[2] = landings[2] + 1
```

> ***Helpful exercises***
>
> - *Write down the code if we landed on 'Free Parking'.*
> - *Write down the code if we landed on 'Mayfair'.*

1.4 Going round the board

Our first go at simulating Monopoly will ignore the community chest, chance cards, and the 'Go To Jail' square. This means that we are simply going round the board. The code in Listing 2 rolls the dice `no_of_rolls` times, and stores the squares that are landed on in the vector `landings`.

Listing 2: Basic Monopoly

```
SimulateMonopoly = function(no_of_rolls) {
  landings = numeric(40)
  #Start at GO
  current = 1
  for(i in 1:no_of_rolls) {
    current = current + RollTwoDice()
    if(current > 40) {
      current = current - 40
    }
    landings[current] = landings[current] + 1
  }
    return(landings)
}
no_of_rolls = 50000
sim = SimulateMonopoly(no_of_rolls)
plot(sim/sum(sim), ylim=range(0.01, 0.04), type='l')
```

> ***Helpful exercises***
>
> - *Carefully type in and run the code in Listing 2. Why do we divide by `sum(sim)`?*
> - *What happens to the plot as you increase `no_of_rolls`?*
> - *What values are the probabilities converging to (we will call this the **baseline** probability)? Add a horizontal line to your plot showing this value.*

1.5 Incorporating Community Chest Cards

There are three community chest squares on the board - squares 3, 18 and 34. In the code below we will just consider square 3. There are sixteen cards in total, hence the probability of drawing any particular card is 1/16. In the code below we will **only implement the first two community chest cards**:

Listing 3: Basic Community Chest function

```
CommunityChest = function(current) {
  goto = current
  u = runif(1)
  if(u < 1/16) {
    goto = 1#Move to Go
  }else if(u < 2/16) {
    goto = 11#Go To Jail
  }
  return(goto)
}
```

This function takes in the `current` position; with probability 1/16 we 'Move to Go', with probability 1/16 we 'Go to Jail' and with probability 14/16 we stay in our current position. We now alter Listing 2 to incorporate the `communityChest` function:

Listing 4: Basic Monopoly

```
SimulateMonopoly1 = function(no_of_rolls) {
  landings = numeric(40)
  #Start at GO
  current = 1
  for(i in 1:no_of_rolls) {
    current = current + RollTwoDice()
    if(current > 40) {
      current = current - 40
    }
    landings[current] = landings[current] + 1
    if(current == 3) {
      cc_move = CommunityChest(current)
      if(cc_move != current){
        current = cc_move
        landings[current] = landings[current] + 1
      }
    }
  }
  return(landings)
}
no_of_rolls = 200000
sim2 = SimulateMonopoly1(no_of_rolls)
plot(sim2/sum(sim2), ylim=range(0.01, 0.04))
```

1.6 What to do

1. Run the simulation function `SimulateMonopoly1`, as given in Listing 4. Include the plot, with baseline probability, in your report, *but do not include any R code.* [14%]

2. You should now add layers of complexity to the `SimulateMonopoly1` function. The final result of this part of the project should be a new version of the `SimulateMonopoly1` function *which you should include in your report.* You should also include any new helper functions that you write (e.g. `RollTwoDice`; see part (h)).

   (a) Add the two other community squares into the `SimulateMonopoly1` code; [+7%]

   (b) Add in 'Go to Old Kent Road' into your `CommunityChest` function; [+7%]

   (c) Square 31 is 'Go To Jail'. Implement this in your main simulation function; [+7%]

   (d) Create a `Chance` function, that implements the first six Chance cards. When you land on a Chance square, call this function; [+10%]

   (e) Add in Community Chest card four; [+10%]

   (f) Add in Chance card 8; [+10%]

   (g) Add in Chance card 7: 'Go back 3 spaces'; [+10%]

   (h) Rolling a double (e.g. a pair of 2's) is special. If you roll a double, you move forward and get to roll the dice again; roll another double, and you get to roll the dice again! But roll a double three times in a row and you go to jail. Write a new helper function to replace `RollTwoDice` to incorporate rolling doubles, and include this new helper function in your report. [+10%]

3. Produce another plot like that from your basic `SimulateMonopoly1` function in part 1, but including the extra layers of complexity you built up in part 2. Comment. [+5%]

4. Other marks are awarded for code presentation (e.g. indentation, use of colour) and style. [+10%]

Some notes

- Submit nicely formatted code. This can printed directly from `RStudio` or from Word. If you use Word, make sure the formatting isn't messed up, and take advantage of highlighting code using bold, colour and indenting.

- You should only include a single simulation function in your report, which has as many of the additional features described in section 2 as you can manage. If you submit more than one simulation function, *you will lose marks.*

- Of course, you should include other helper functions – for example, `Chance`, `CommunityChest`, `RollTwoDice`.

- If you submit a simulation function that produces an error, i.e. it doesn't run without crashing, you may get zero marks.

- Don't submit your plot commands. Just submit the relevant R functions.

- State clearly how many simulations you used. Your final results should probably be based on at least 100,000 simulations, but more if your computer can manage it!

- Remember to label all axes and put your names and group number on the front cover of your work.

2 Sticker album

2.1 Preliminaries

Last year's *Panini* sticker album for the FIFA '18 World Cup football finals had a whopping 682 stickers to collect, including 40 awesome glitter stickers! Each packet of stickers cost 80p and contained 5 stickers. The obvious question to ask is:

"How much does it cost, on average, to fill a Panini World Cup '18 sticker album?"

Look at the code in Listing 5 below.

Listing 5: Album filling code

```
stickers = 1:50 #Sticker sequence.  Change later.
for(i in 1:10000) {#Insert comments here
  packet = sample(1:50, 5, replace=TRUE)#Insert comments here

  #Insert comments here
  for(j in 1:5){
    stickers[packet[j]] = 0
  }

  #Insert comments here
  if(sum(stickers)==0) {
    break
  }
}
i
```

Helpful exercises

- *Run the code lots of times and try to figure out what's going on. Talk to the other members of your group - it's vital that you try to understand the code at this point!*
- *What does the* `break` *command do?*

2.2 What to do

The ultimate goal is to work out, on average, how much it will cost to fill the sticker album.

1. You should try to incorporate the code in Listing 5 into a function which returns the number of packets needed to fill the album. *You should include this function in your report*, making sure you annotate the main code at the points where it says "*Insert comments here*" in Listing 5; [40%]
2. You should call your function in 1 lots of times, using a `for` loop, storing your results in a vector. *There is nothing to include in your report here*;
3. Explore the resulting vector, and think about how you could use this to investigate the cost of filling the sticker album. Use graphical and numerical summaries, including anything you think might be relevant in your report. *It is not necessary to include any R code here.* [+50%]
4. Other marks are awarded for code presentation and style. [+10%]

Appendix: Additional information about Monopoly

Community Chest Cards

There are three community chest areas on the board (see Table 1). In total, there are 16 community chest cards.

1. Advance to Go;
2. Go to jail;
3. Go to Old Kent Road;
4. Take a Chance card instead;

Chance Cards

A Chance card is most likely to move players. There are three chance areas on the board (see Table 1). There are 16 chance cards in total, of which eight cards move the player:

1. Advance to Go;
2. Advance to Trafalgar Square;
3. Advance to Pall Mall;
4. Go directly to Jail;
5. Take a trip to Marylebone Station
6. Advance to Mayfair
7. Go back 3 spaces;
8. Advance token to nearest Utility. The utility squares are the water works and the electric company.

Square Number	Name	Square Number	Name
1	Go	11	Jail
2	Old Kent Road	12	Pall Mall
3	Community Chest	13	Electric Company
4	WhiteChapel Road	14	Whitehall
5	Income tax	15	Northumberland Avenue
6	King's Cross Station	16	Marylebone station
7	The Angel Islington	17	Bow Street
8	Chance	18	Community Chest
9	Euston Road	19	Marlborough Street
10	Pentonville Road	20	Vine Street
21	Free Parking	31	Go To Jail
22	Strand	32	Regent Street
23	Chance	33	Oxford Street
24	Fleet Street	34	Community Chest
25	Trafalgar Square	35	Bond Street
26	Fenchurch Street Station	36	Liverpool St Station
27	Leicester Square	37	Chance
28	Coventry St	38	Park Lane
29	Water Works	39	Super Tax
30	Piccadilly	40	Mayfair

Table 1: *Monopoly squares with associated square numbers*