(Exercise 8) You are going to model how to play darts. If you want a simpler version then solve the problem using only the sets A and B described overleaf, and not C. Details are in Application 3.7 and the exercise itself. Please do feel free to ask if you're unsure about the setup. If you're not sure what the game of darts is, then just do a quiz web image search for a 'dartboard' it's pretty obvious from there.

In this exercise you will be called upon to write a *recursive* function. Please look in the section on 'Functions' in the original R Handout for some examples of recursive functions. You may find it useful to copy out the recursive factorial function from that handout to get you started before you attempt the coding parts of this exercise.

When trying to maximize amongst a set of choices $a \in \mathcal{A}$ you may find it useful to think about what the following code does:

```
v <- rep(0, 10)
for (i in 1:10) {
    v[i] = sin(i)
}
V <- max(v)
cat(V)
```

Some more specific help for the exercise may also include:

- When formulating the action set $\mathcal{A}_x$ of available targets when in state x you may find **is**.element a useful function, as well as **union**. Don't forget to not permit going past 0, or landing on zero using $a \in A \setminus B$. It may help you to see what the code A[A<4] does.
- As a reminder for loops, you can use the **for**(a in S) construction to ask R to loop a over the elements of S.
- You may like to see out what 1*(x>0) does, although **as.numeric**(x>0) is perhaps better practice.
- When implementing an optimal value function $V_n(\cdot)$ as a function in code, you will find that you need to cope with the case $n = 0$ (i.e. the boundary case) separately at the beginning of the function. Hence lines like:

  ```
  if(n==0){
    if(x==0){
      return(7)
    } else {
      return(14)
    }
  }
  ```

  might be very helpful.

The R handout from the start of the term contains many useful examples, including some of those above. As we study more complicated problems in the final few weeks you may find the later exercises in the handout very useful.

Exercise 8 (Theon)**.** *How to play darts, optimally.* The setup for this problem was explained in Application 3.7. Go and read that before starting. Here's a reminder of the optimality equation to use, and the region types:

$$V_n(x) = \max_{a \in \mathcal{A}_x} \left[R(a, x) + V_{n-1}(x - a) \right].$$

- Type A regions with values: $0, 1, 2, 3, \ldots, 19, 20, 25$
- Type B regions with values: $2, 4, 6, \ldots, 38, 40, 50$
- Type C regions with values: $3, 6, 9, \ldots, 54, 57, 60$

(a) (in R) Create vectors A, B and C, which hold the values of regions in their respective types. Then write code to determine the action set $\mathcal{A}_x$ of permitted actions. Create a function called 'actions' with input A, B and C and output $\mathcal{A}_x$. *Note: you cannot take actions which would move you to a negative state.*

(b) Create a function, f with inputs A, B, C, n, and x which works out the value of $V_n(x)$. The function can be recursive, and thus use f itself when you want to know $f(A, B, C, n-1, \cdot)$. Hint: the code in your function should mainly be finding the values of the right-hand side of the OE, then selecting the maximum. *Part (b) is the main part of the exercise.*

(c) Find the optimal value $f_2(x)$ for all $x \in \{10, 11, \ldots, 20\}$. Is the answer what you expected?

(d) Use your code to find the smallest three values of $x > 1$ such that you cannot reach 0 in fewer than 3 turns.

(e) Suppose now that you start from $m = 104$ (and $n = 3$). Can you use your code to find the minimal number of turns required, and also what the optimal first target is?

Also interersting are: 97, 107, 95

(f) Do you have any intuition of how to play the game optimally? How would your intuition get on if you started with $n = 3$ from $x = 59$ or $x = 69$?

(g) Why does it take so long to evaluate $V_4(138)$, $V_5(188)$ and $V_6(238)$? (The last one might take a while, you don't have to wait for it to finish!)

(h) How would you modify your code if you weren't as skilled, but instead when aiming for a double had a 50% chance of missing the board entirely, and 50% chance of hitting your target?