R Programming

Problem Set

Table of contents

1 Q1

Write an R function `print_hollow_square()` to print a square pattern with `*` characters, but make it *hollow*.

- Example
  - input: `5` (size of the square)
  - output:

```
*****
*   *
*   *
*   *
*****
```

2 Q1 - Solution *ver1*

```r
print_hollow_square = function(size) {
  for (i in 1:size) {
    cat ("*")
  }
  cat ("\n")

  for (i in 2:(size-1)) {
    for (j in 1:size) {
      if (j == 1 || j == size) {
        cat("*")
      } else {
        cat(" ")
      }
    }
    cat("\n")
  }

  for (i in 1:size) {
    cat ("*")
  }
  cat ("\n")
}

```

```r
print_hollow_square(5)
```

```
*****
*   *
*   *
*   *
*****
```

3 Q1 - Solution *ver2*

```r
print_hollow_square = function(size) {
  cat (rep("*", size), "\n", sep = "")

  for (i in 2:(size-1)) {
    for (j in 1:size) {
      if (j == 1 || j == size) {
        cat("*")
      } else {
        cat(" ")
      }
    }
    cat("\n")
  }

  cat (rep("*", size), "\n", sep = "")
}

print_hollow_square(5)
```

```
*****
*   *
*   *
*   *
*****
```

4 Q1 - Solution *ver3*

```r
print_hollow_square = function(size) {
  for (i in 1:size) {
    for (j in 1:size) {
      if (i == 1 || i == size || j == 1 || j == size) {
        cat("*")
      } else {
        cat(" ")
      }
    }
    cat("\n")
  }
}

print_hollow_square(5)
```

```
*****
*   *
*   *
*   *
*****
```

5 Q2

Write an R function `roll_dice()` that simulates rolling a fair six-sided die n times, and returns the count of each face value.

- Example
  - input: `100`
  - output: A named vector with counts for each face value (e.g., `c('1' = 14, '2' = 19, '3' = 15, '4' = 18, '5' = 17, '6' = 17)`)

6 Q2 - Solution

```r
roll_dice = function(n) {
  rolls = sample(1:6, n, replace = TRUE)
  table(rolls)
}
roll_dice (100)
```

```
rolls
 1  2  3  4  5  6 
16 17 16 21 13 17 
```

7 Q3

Write an R function `second_largest()` to find the second largest number in a vector.

- Example
  - input: `c(1, 3, 4, 5, 0, 2)`
  - output: `4`

8 Q3 - Solution

```r
second_largest = function(numbers) {
  if (length(numbers) < 2) {
    return(NULL)
  }

  sorted_numbers = sort(numbers, decreasing = TRUE)
  return(sorted_numbers[2])
}

second_largest(c(1, 3, 4, 5, 0, 2))
```

```
[1] 4
```

9 Q4

Write an R function `square_or_cube()` that takes a numeric vector as input and returns a new vector with the square of each number if it is even and the cube of each number if it is odd.

- Example
    - input: `c(1, 2, 3, 4, 5)`
    - output: `c(1, 4, 27, 16, 125)`

10 Q4 - Solution *ver1*

```r
square_or_cube = function(numbers) {
  result = c()
  for (num in numbers) {
    if (num %% 2 == 0) {
      result = c(result, num^2)
    } else {
      result = c(result, num^3)
    }
  }
  return (result)
}

square_or_cube(c(1, 2, 3, 4, 5))
```

```
[1]   1   4  27  16 125
```

11 Q4 - Solution *ver2*

```r
square_or_cube = function(numbers) {
  result = ifelse (numbers %% 2 == 0, numbers^2, numbers^3)
  return (result)
}

square_or_cube(c(1, 2, 3, 4, 5))
```

```
[1]   1   4  27  16 125
```

12 Q5

Write an R function `is_prime_number()` that takes an integer `n` as input and returns `TRUE` if `n` is a prime number, and `FALSE` otherwise. A prime number is a natural number greater than 1 that has no positive divisors other than 1 and itself.

- Example:
  - input: `7`
  - output: `TRUE`
- Example:
  - input: `10`
  - output: `FALSE`

13 Q5 - Solution *ver1*

```r
is_prime_number = function(n) {
  if (n <= 1) {
    return(FALSE)
  }
  if (n == 2) {
    return(TRUE)
  }
  for (i in 2:floor(sqrt(n))) {
    if (n %% i == 0) {
      print(i)
      return(FALSE)
    }
  }
  return(TRUE)
}
is_prime_number(7)
```

```
[1] TRUE
```

```r
is_prime_number(10)
```

```
[1] 2
```

```
[1] FALSE
```

14 Q5 - Solution *ver2*

```r
is_prime_number = function(n) {
  if (n <= 1) {
    return(FALSE)
  }
  if (n == 2) {
    return(TRUE)
  }
  return (all(ifelse (n %% 2:floor(sqrt(n)) == 0, FALSE, TRUE)))
}
is_prime_number(7)
```

```
[1] TRUE
```

```r
is_prime_number(10)
```

```
[1] FALSE
```

15 Q5 - Solution *ver3*

```r
is_prime_number = function(n) {
  if (n <= 1) {
    return(FALSE)
  }
  if (n == 2) {
    return(TRUE)
  }
  return (all(n %% 2:floor(sqrt(n)) != 0))
}
is_prime_number(7)
```

```
[1] TRUE
```

```
is_prime_number(10)
```

```
[1] FALSE
```