Problem 3: Merchant's guide to the galaxy:

Explanations and assumptions:

In order to solve this problem, I have selected R as programming language as this the language in what I have my most experience. I wrote my function slightly more general so that it not only works with the given input from the excercise. Of course, the function could be even more general, but this was difficult given the small amount of input information.

My assumptions were:

- There are only four intergalactic units: glob, prok, pish and tegj.
- There are only three goods of interest: silver, gold and iron.
- The input for the parameters *silver_input*, *gold_input* and *iron_input* has to start with two intergalactic units and the fifth word has to be a numeric value.
- Only two types of questions are allowed starting with either *how much is* or *how many Credits is*.
- The first question type has to contain four intergalactic units after *how much is*.
- The second question type has to contain two intergalactic units after *how many Credits is*, followed by the name of one of the three goods.

Below, you can find the code as well as the input and output from the exercise. There exists one standard function in R that made the whole task a lot easier: *as.roman*. This function can convert objects into roman numerals. Afterwards, I was able to use the powerful function *as.numeric* to convert the roman numeral to an arabic numeral. At the beginning, I was not sure if this could be considered as cheating. But since you wrote in the exercise that I am free to use *„build systems and libraries the same way you would build real-world software“*, I thought this should be fine. In the real-world, I would use this function as well as this would safe me a lot of time.

I hope that my code quality fulfills your expectations. I have tried to make the code easy readable and understandable. I could make the code shorter and more efficient but this would make it also harder to comprehend. In addition, I have tried to consider all possibilities of invalid input for my function and I made sure that my function returns an understandable error message in that case.

```
Intergalactic_Unit_Converter = function(question, glob = "I", prok = "V",
                                        pish = "X", tegj = "L",
                                        silver_input = "glob glob Silver is 34 Credits",
                                        gold_input = "glob prok Gold is 57800 Credits",
                                        iron_input = "pish pish Iron is 3910 Credits") {

  # Split elements of the character object into substrings:
  split_question = strsplit(question, " ")
  split_silver = strsplit(silver_input, " ")
  split_gold = strsplit(gold_input, " ")
  split_iron = strsplit(iron_input, " ")

  # Handling invalid input:
  stopifnot(is.character(question),
            length(split_question[[1]]) >= 7,
            is.character(silver_input),
            is.character(gold_input),
            is.character(iron_input),
            length(split_silver[[1]]) >= 5,
            length(split_gold[[1]]) >= 5,
            length(split_iron[[1]]) >= 5,
```

```
          all(split_silver[[1]][1:2] %in% c("glob", "prok", "pish", "tegj")),
          all(split_gold[[1]][1:2] %in% c("glob", "prok", "pish", "tegj")),
          all(split_iron[[1]][1:2] %in% c("glob", "prok", "pish", "tegj")),
          !is.na(as.numeric(split_silver[[1]][5])),
          !is.na(as.numeric(split_gold[[1]][5])),
          !is.na(as.numeric(split_iron[[1]][5])),
          glob  %in% c("I", "V", "X", "L", "C", "D", "M"),
          prok  %in% c("I", "V", "X", "L", "C", "D", "M"),
          pish  %in% c("I", "V", "X", "L", "C", "D", "M"),
          tegj  %in% c("I", "V", "X", "L", "C", "D", "M"))


# Convert intergalactic units into the roman numerals:
convert_silver = paste(eval(as.name(split_silver[[1]][1])),
                       eval(as.name(split_silver[[1]][2])), sep = "")
convert_gold = paste(eval(as.name(split_gold[[1]][1])),
                     eval(as.name(split_gold[[1]][2])), sep = "")
convert_iron = paste(eval(as.name(split_iron[[1]][1])),
                     eval(as.name(split_iron[[1]][2])), sep = "")

# Convert an object of class "character" into an object of class "roman"
# with the as.roman function:
roman_number_silver = as.roman(convert_silver)
roman_number_gold = as.roman(convert_gold)
roman_number_iron = as.roman(convert_iron)

# Convert an object of class "roman" into an object of class "numeric"
# with the as.numeric function:
arabic_number_silver = as.numeric(roman_number_silver)
arabic_number_gold = as.numeric(roman_number_gold)
arabic_number_iron = as.numeric(roman_number_iron)

# Calculate the silver, gold and iron value:
silver_value = as.numeric(split_silver[[1]][5]) / arabic_number_silver
gold_value = as.numeric(split_gold[[1]][5]) / arabic_number_gold
iron_value = as.numeric(split_iron[[1]][5]) / arabic_number_iron

# Handling the first question type:
if (grepl("how much is", question) == TRUE) {

  # Handling invalid input:
  stopifnot(all(split_question[[1]][4:7] %in% c("glob", "prok", "pish", "tegj")))

  # The same procedure as above:
  convert_question = paste(eval(as.name(split_question[[1]][4])),
                           eval(as.name(split_question[[1]][5])),
                           eval(as.name(split_question[[1]][6])),
                           eval(as.name(split_question[[1]][7])), sep = "")

  roman_number_question = as.roman(convert_question)
  arabic_number_question = as.numeric(roman_number_question)

  # Generate the output:
  output = paste(split_question[[1]][4], split_question[[1]][5],
```

```
                  split_question[[1]][6],  split_question[[1]][7], "is",
                  arabic_number_question, sep = " ")

    return(output)
  }

  # Handling the second question type:
  if (grepl("how many Credits is", question) == TRUE) {

    # Handling invalid input:
    stopifnot(all(split_question[[1]][5:6] %in% c("glob", "prok", "pish", "tegj")))

    # The same procedure as above:
    convert_question = paste(eval(as.name(split_question[[1]][5])),
                             eval(as.name(split_question[[1]][6])), sep = "")
    roman_number_question = as.roman(convert_question)
    arabic_number_question = as.numeric(roman_number_question)

    # Generate the output for Silver:
    if (split_question[[1]][7] == "silver" | split_question[[1]][7] == "Silver") {

      output = paste(split_question[[1]][5], split_question[[1]][6],
                     split_question[[1]][7], "is",
                     arabic_number_question * silver_value, "Credits", sep = " ")

      return(output)
    }

    # Generate the output for Gold:
    if (split_question[[1]][7] == "gold" | split_question[[1]][7] == "Gold") {

      output = paste(split_question[[1]][5], split_question[[1]][6],
                     split_question[[1]][7], "is", arabic_number_question * gold_value,
                     "Credits", sep = " ")

      return(output)
    }

    # Generate the output for Iron:
    if (split_question[[1]][7] == "iron" | split_question[[1]][7] == "Iron") {

      output = paste(split_question[[1]][5], split_question[[1]][6],
                     split_question[[1]][7], "is", arabic_number_question * iron_value,
                     "Credits", sep = " ")

      return(output)
    } else {
      return("Error: wrong input format.")
    }

  } else {
    return("I have no idea what you are talking about")
  }
}
```

```
Intergalactic_Unit_Converter(question = "how much is pish tegj glob glob ?")

## [1] "pish tegj glob glob is 42"

Intergalactic_Unit_Converter("how many Credits is glob prok Silver ?")

## [1] "glob prok Silver is 68 Credits"

Intergalactic_Unit_Converter("how many Credits is glob prok Silver ?",
                             silver_input = "pish pish silver is 500")

## [1] "glob prok Silver is 100 Credits"

Intergalactic_Unit_Converter("how many Credits is glob prok Gold ?")

## [1] "glob prok Gold is 57800 Credits"

Intergalactic_Unit_Converter("how many Credits is glob prok Iron ?")

## [1] "glob prok Iron is 782 Credits"

Intergalactic_Unit_Converter("how much wood could a woodchuck chuck if a woodchuck
                             could chuck wood ?")

## [1] "I have no idea what you are talking about"
```