STAT2005 Programming Languages for Statistics
Exercise for Chapter 1

1. Using `rep()` and `seq()` as needed to create the following vectors. (The use of `c()` function is prohibited in this question.)

(a) 5 10 15 20 25 30 35 40 45 50 seq(5, 50, by=5)

(b) 6 6 5 5 4 4 3 3 2 2 1 1 rep(seq(6, 1, by = -1), each = 2)

(c) 1 2 3 4 5 3 4 5 6 7 5 6 7 8 9 7 8 9 10 11 rep(seq(1,5), 4) + rep(seq(0, 6, by=2), each=5)
(+0 +2 +4 +6)

(d) 2 4 6 5 7 9 8 10 12 11 13 15 (Hint: `seq()` is not needed) alt: 1:20 - rep(seq(0,9,3), each=4)
(+0 +3 +6 +9)
rep(seq(2,6,by=2), 4) + rep(seq(0,9,by=3), each=3)
↳ means ugly `:`
alt: 1:12 + rep(1:3, 4)

2. The information of overseas exchange participants is collected. The information collected include

- `school` - school of participant
- `destination` - destination of overseas exchange participant
- `age` - age of participants

(a) The data are stored in a data frame named `participants` as shown below.

```
> participants
  school destination age
1   CUHK     Austria  18
2   <NA>       Japan  20
3   HKBU       Korea  22
4  HKUST     Austria  NA
5   CUHK      Sweden  20
```

Write the R codes to create this data frame.

(b) It is found that "Australia" is mistyped as "Austria" in the data. Write the R codes to replace all "Austria" by "Australia" in `participants`.

(c) Write the R codes to change "<NA>" in the second row of `school` column in `participants` into "HKU".

(a):
```
participants <- data.frame (
    school = c( "CUHK", NA, "HKBU", "HKUST", "CUHK"),
    destination = c("Austria", "Japan", "Korea", "Australia", "Sweden")
    age = c( 18, 20, 22, NA, 20)
)
```

(b) ifelse (participants$destination == "Austria", "Australia", participants$destination)

(c) ifelse (is.na(participants$school), "HKU", participants$school)
↳ needs to assign: participants$destination <- ifelse (...)

alt: (b)

participants$ destination ← factor(participants$ destination)
levels(participants $ destination) ← C("Australia", "Japan", Korea", "Sweden")

3. (a) Write the R codes to create the following ~~object named~~ `list1`.

```
> list1
[[1]]
[1] "apple"  "banana"

[[2]]
[1] 1 2 3

[[3]]
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    6    7    8    9   10
```

(b) Write the R codes to change `list1` to a long vector.

(c) After ~~finishing (b), write the~~ output of the following codes (without using computer):

i. `class(list1)`
ii. `mode(list1[3])`
iii. `class(mode(list1))`
iv. `mode(factor(c(list1[8],list1[9])))`

(a): list1 ← list(c("apple","banana"), c(1, 2, 3), matrix(1:10, nrow=2, byrow=T))

(b) list1 ← unlist(list1)

(c) i. "character"
ii. "character"
iii. "character"
iv. "numeric"

STAT2005 Programming Languages for Statistics
Exercise for Chapter 2

1. Mary is living in an integer-valued one-dimensional space. She is initially at the origin at $x = 0$. She can either move to the left or right by 1 or 2 units in each second with equal probabilities. E.g. If Mary is at 0, she can move to -2, -1, 1, 2, in a second with equal probabilities.

(a) Using a random seed `2005`, write R codes to simulate Mary's movements during the first 120 seconds. Draw the path over time.

(b) Plot two red dashed horizontal lines at the maximum and minimum of the path.

2. (a) Using a random seed `2005`, generate 1,000 pseudorandom numbers from $X \sim N(3,4)$, store them in a vector named `x`.

(b) Generate 2,000 pseudorandom numbers from $Y \sim N(1,4)$, store them in a vector named `y`. Assume that the population mean and standard deviations of X and Y are unknown to us, but we know that their standard deviations are the same. Write R codes to find the pooled standard deviation of `x` and `y` and store it into a variable named `PooledSD`. The formula of pooled standard deviation is

$$s_p = \sqrt{\frac{(n_1-1)s_1^2+(n_2-1)s_2^2}{n_1+n_2-2}},$$

where n_1, n_2 are the sample size of `x` and `y`, s_1^2 and s_2^2 are the sample variance of `x` and `y`.

(c) We are interested to perform a two-sample t-test for `x` and `y`.

$$H_0: \mu_1 = \mu_2 \text{ vs. } H_1: \mu_1 \neq \mu_2,$$

where μ_1, μ_2 are the unknown population means of X and Y. Write R codes to find the following t-statistics and compute the corresponding critical value at 95% significance level. The t-statistics is given by

$$t = \frac{(\bar{x}-\bar{y})}{s_p\sqrt{\frac{1}{n_1}+\frac{1}{n_2}}} \sim t_{n_1+n_2-2},$$

where t_{df} denotes the t-distribution with df degrees of freedom.

3. The file `ex2_q3.dat` stores the data of the GDP (Gross Domestic Product) for 26 countries (from "A" to "Z") in the first and second year. The variables are

`Country`: Country label from "A" to "Z";

`gdp1`: GDP of the first year;

`gdp2`: GDP of the second year;

`Region`: Each country belongs to one of the four regions "East", "South", "West" or "North".

(a) Write R codes to read `gdp.dat` as a `data.frame` object named `data`. Display the first 6 data.

(b) Write R codes to find the mean of `gdp1` in each `region`.

(c) Using the `by()` function, compute the sum of values in column `gdp1` and `gdp2`.

(d) Draw a scatter plot of `gdp1` and `gdp2` to show the relationship between GDP in the first and second year. Do you find any linear relation between the two?

1. (a).

```
set.seed(2025)
moves <- sample(c(-2,-1,1,2), size=200, replace=T, prob=c(0.25,0.25,0.25,0.25))
pos <- cumsum(c(0, moves))
pos <- as.ts(pos)
plot(pos)
```

(b)

```
max_pos <- max(pos)
min_pos <- min(pos)
abline(h=max_pos, col="red", lty="dashed")
abline(h=min_pos, col="red", lty="dashed")
```

2(a)

```
set.seed(2025)
x <- rnorm(1000, 3, 4)
```

(b)

```
y <- rnorm(2000, 1, 4)
n1 <- length(x)
n2 <- length(y)
s1_2 <- var(x)
s2_2 <- var(y)
Pooled SD <- sqrt((n1-1)*s1_2 + (n2-1)*s2_2) / (n1+n2-2))
```

(c)

```
x_mean <- mean(x)
y_mean <- mean(y)
t <- (x_mean - y_mean) / (Pooled SD * sqrt(1/n1 + 1/n2))
abs(t) > qt(0.975, n1+n2-2))
```

3. (a)

```
data <- read.table("gdp.dat", header=T)
head(data)
```

(b)

```
by(data$gdp1, data$region, mean)
```

for vector, use tapply Region //, use by is also OK

(c)

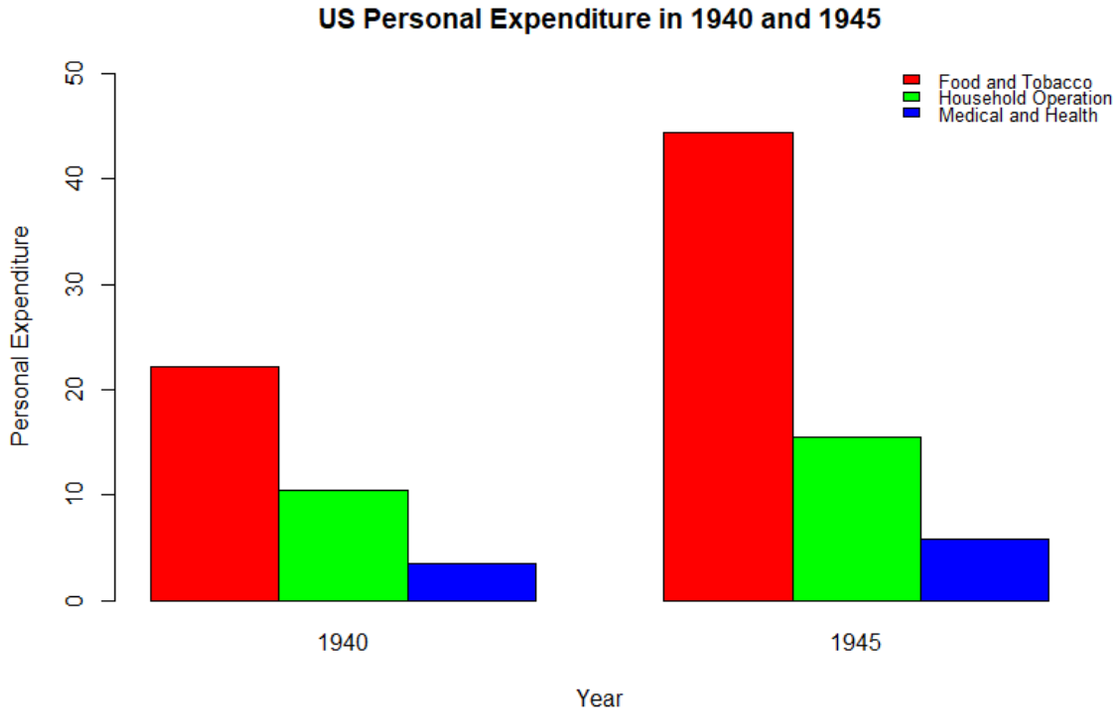
```
by(data[,c(2,3)], data$region, colSum)
rel).plot(data$gdp1, data$gdp2)
```

STAT2005 Programming Languages for Statistics
Exercise for Chapter 3

1. Consider the built-in data frame `USPersonalExpenditure`.

(a) Rename `USPersonalExpenditure` into a short form `USPE`. Plot a vertical bar chart to show the US personal expenditure in *Food and Tobacco*, *Household Operation* and *Medical and Health* in 1940. Set the relative font size of the axis names (bar labels) to 0.8, and the range of y-axis to (0,25). You are not required to add title and x-, y-axis labels.

(b) Using the data in `USPE`, plot the bar chart as follows.



Note that the legend should not overlap with the bars. Search for the arguments in `args.legend` to adjust the font size and positon of the legend to avoid the overlap if necessary. Set the bar colors to `rainbow(3)`.

2 The file `ex3_q2.csv` stores the data of a gaming competition in 2023. The competition consists of two rounds. The data in each row represent a team. The variables are

`R1`: Scores of the team in Round 1;

`R2`: Scores of the team in Round 2.

(a) Write R codes to read `ex3_q2.csv` as a `data.frame` object named `data`. Plot two boxplots for each column in `data`.

(b) NRG is a gaming team participating in this competition. They scored 36 in Round 1, and 41 in Round 2. Plot NRG's scores in each round on the two plots as points respectively.

Referring to the plots only, which round did NRG perform better?

(c) The total score of the n^{th} team is given by

$$total_n = \frac{R_{1,n}+R_{2,n}}{2},\ 1 \le n \le 20$$

Plot a normal Q-Q plot for $total$ and add a red reference line to check whether $total$ fits in normal distribution.

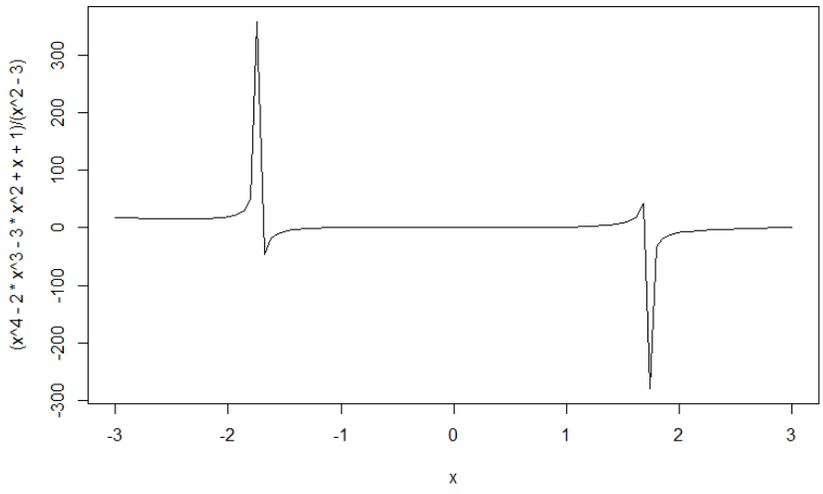
3. (a) Risky tried to write R codes to plot the following curve on $[-3,3]$.

$$y = \frac{x^4 - 2x^3 - 3x^2 + x + 1}{x^2 - 3}$$

This curve has two asymptotes $x = \pm\sqrt{3}$. By executing

```
curve((x^4-2*x^3-3*x^2+x+1)/(x^2-3),-3,3)
```

in R, Risky got the following graph:



However, the curve around $x = \pm\sqrt{3}$ is incorrectly displayed. Please help Risky to plot the correct curve. Set the limit on x-axis to $(-3,3)$ and y-axis to $(-250,250)$ respectively. (Hints: use `seq()`, `plot()`, and `lines()` to plot three parts of the curve one by one.)

(b) Plot two red asymptotes $x = \pm\sqrt{3}$ to the curve.

(c) Using `segments()` function, draw a triangle with vertex $(-1,100)$, $(-1,200)$ and $(0,200)$.

(d) Draw a blue-filled triangle with vertex $(0,100)$, $(1,100)$ and $(1,200)$.

1.

(a)

USPE ← USPersonalExpenditure

barplot(USPE$1940[1:3], cex=0.8, ylim=c(0,25))

cex.names=0.8

USPE[1:3,] ↑ should work, but in fact it is not data from

(b)

barplot(USPE[1:2,1:3], legend.txt = names(USPE$1940)[1:3], args.legend = list(bty="n"), col = rainbow(3))

beside=T

```
# Q1b
b<-barplot(
  USPE[1:3,1:2],
  col=rainbow(3),
  ylim=c(0,50),
  beside=T,
  legend=T,
  args.legend=list(x="topright",bty="n",inset=c(-0.08, -0.02),cex=0.8),
  # x="topright" and inset adjusts the position
  # cex adjusts the font size
  xlab="Year",
  ylab="Personal Expenditure",
  main="US Personal Expenditure in 1940 and 1945"
)
```

2. (a)

data ← read.table("ex3-q2.csv", header=T, sep=",")

~~par(mfrow=c(1,2))~~

~~boxplot(data$R1)~~

~~boxplot(data$R2)~~

boxplot(data)

(b)

for bxt plot, x is 1, 2, ...

points(x=1, 36, pch=19)

points(x=2, 41, pch=19)

```
# Q3
curve((x^4-2*x^3-3*x^2+x+1)/(x^2-3),-3,3)

# Q3a
x1 <- seq(-3,-sqrt(3)-0.001,by=0.001)
x2 <- seq(-sqrt(3)+0.001,sqrt(3)-0.001,by=0.001)
x3 <- seq(sqrt(3)+0.001,3,by=0.001)
y1 <- (x1^4-2*x1^3-3*x1^2+x1+1)/(x1^2-3)
y2 <- (x2^4-2*x2^3-3*x2^2+x2+1)/(x2^2-3)
y3 <- (x3^4-2*x3^3-3*x3^2+x3+1)/(x3^2-3)
plot(x1,y1,type="l",xlim=c(-3,3),ylim=c(-250,250))
lines(x2,y2)
lines(x3,y3)

# Q3b
abline(v=sqrt(3),lty=2,col="red")
abline(v=-sqrt(3),lty=2,col="red")

# Q3c
segments(-1,200,0,200)
segments(-1,100,0,200)
segments(-1,100,-1,200)

# Q3d
polygon(c(0,1,1),c(100,100,200),col="blue")
```

(c)

~~total ← (data$R1 + data$R2)/2~~

~~qqnorm(total)~~

~~qqline(total, col="red")~~

3. (a)

fx ← function(x) {

(x^4 - 2*x^3 - 3*x^2 + x + 1) / (x^2 - 3)

}

plot (-3, -√3):

xs ← seq(-3, -sqrt(3), by = 0.001)

ys ← fx(xs)

plot(xs, ys, type="l", xlim = c(-3,3), ylim = c(-250, 250))

(b)

abline(v = -sqrt(3), col="red", lty=dashed)

abline(v = sqrt(3), col="red", lty=dashed)

(c).

segments(-1, 100, -1, 200, 0, 200)

(d)

polygon(c(0,1,1), c(100,100,200), col="blue")

STAT2005 Programming Languages for Statistics
Exercise for Chapter 4

1. (a) Initialize a graph without the origin and border, set the x- and y-axis on the range from 0 to 100. Add a square with vertex $(0,0)$, $(0,100)$, $(100,0)$, and $(100,100)$.

(b) Given four points $A_1 = (0,100), B_1 = (100,100), C_1 = (100,0), D_1 = (0,0)$.

Then, we have the recursive relation

$$A_n = \frac{A_{n-1}+B_{n-1}}{2}, B_n = \frac{B_{n-1}+C_{n-1}}{2}, C_n = \frac{C_{n-1}+D_{n-1}}{2}, D_n = \frac{D_{n-1}+A_{n-1}}{2}, n > 2.$$

Write R codes to add squares $A_1B_1C_1D_1, A_2B_2C_2D_2, A_3B_3C_3D_3, \dots\dots, A_{100}B_{100}C_{100}D_{100}$ on the same graph.

2. A number is a monodigit if it is a positive integer consists of a single repeated digit only, e.g. $2, 33, 444, 5555$.

(a) Write function `checkmono(x)` to check whether a number is a monodigit. `checkmono(x)` returns the repeated digit if `x` is a monodigit, otherwise returns 0.

(b) Using `checkmono()`, write function `mono(n)` to return the sum of digits of monodigits between 1 to n. For example, the monodigits between 1 to 11 are 1, 2, 3, 4, 5, 6, 7, 8, 9 and 11, `mono(11)` returns 47 because $1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 1 + 1 = 47$.

(c) Find the sum of digits of monodigits between 1 to $100{,}000$.

(d) Risky thinks the approach in part (b) is too slow. Instead of checking every number, he wants to improve the time complexity of the program by enumerating all monodigits between 1 to n. Using Risky's approach, write function `mono2(n)` to return the sum of digits of monodigits between 1 to n.

3. Apple City is a city which use apples as badges. This city has n citizens. The more apples a citizen has, the higher rank s/he gets. The number of apples of each citizen has are stored in a vector `apple`, i.e. `apple[1]` corresponds to citizen 1, `apple[2]` corresponds to citizen 2, …… , `apple[n]` corresponds to citizen n.

(a) The government gives apples to or takes apples away from citizens whenever necessary. For each action, it add the number of apples from citizen x to citizen y by integer m, where x, y are integers and $1 \le x \le y \le n$. Note that m can be negative and no more apples can be taken away when a citizen does not have any apples. Without using loops, write function `modify(x,y,m)` to add every value from `apple[x]`, `apple[x+1]`,…, `apple[y-1]`, `apple[y]` by m.

(b) A citizen is 'good ' if the number of apples s/he has is within a particular range. Write function `count(lower,upper)` to count the number of 'good ' citizens in Apple City, where `lower` and `upper` represents the lower and upper bound of the range respectively.

1.(a)

```
plot(0,0, type="n", xlim=c(0,100), ylim=c(0,100), bty="n", xlab="", ylab="")
polygon(c(0, 0, 100, 100), c(0, 100, 0, 100))
```

(b)

```
ax<-0; ay<-100; bx<-100; by<-100
cx<-100; cy<-0; dx<-0; dy<-0

for (i in 1:100) {
    polygon(c(ax, bx, cx, dx), c(ay, by, cy, dy))
    ax_tmp<-(ax+bx)/2;
    ....
}
```

2(a)

```
checkmono <- function(x) {
    if (x <= 0) {
        return(F)
    }
    digit <- x%%10
    while (x >= 10) {
        if (x%%10 != digit) {
            return(F)
        }
        x <- x%/%10
    }
    return(T)
}
```

(b)

```
mono <- function(n) {
    monodigits <- c()
    for (i in 1:n) {
        if (checkmono(i)) {
            monodigits <- c(monodigits, i)
        }
    }
    return(sum(monodigits))
}
```

(c)

```
mono(100000)
```

(d)

```
mono2 <- function(n) {
    mds <- c()
    base <- 1
    repeat {
        for (scale in 1:9) {
            tmp <- base * scale
            if (tmp > n) {
                return(sum(mds))
            }
            mds <- c(mds, tmp)
        }
        base <- base *10 +1
    }
}
```

```r
# Q3a
apple <- 1:10 # declare a global vector apple to test this function
# Answer:
modify <- function(x,y,m){
  apple[x:y] <<- apple[x:y]+m
  # use <<- to modify global variable
  apple[apple<0] <<- 0
  # change negative values to 0
}

# Q3b
count <- function(lower,upper){
  return(sum(apple>=lower&apple<=upper))
  # return (length(apple[apple>=lower&apple<=upper])) is also ok
}
```

3(a) note: in function to modify global var, use <<—

```
modify <- function(x, y, m) {
    apply_idx <- ifelse((1:n) >= x && (1:n) <= y, 1, 0)
    apply_amount <- apply_idx * m
    apple <<- apple + apply_amount
    return(ifelse(apple < 0, 0, apple))
}
```

(b)

```
count <- function(lower, upper) {
    mask <- ifelse(apple >= lower && apple <= upper, 1, 0)
    return(sum(mask))
}
```

STAT2005 Programming Languages for Statistics
Exercise for Chapter 5

1. (a) Risky has created a new non-negative integer sequence called Risky Sequence. Risky Sequence $\{R(n)\}$ is defined by

$$R(n) = \begin{cases} 1, & \text{for } n = 0 \text{ or } 1, \\ 4R(n-1) - 3R(n-2), & \text{for odd } n > 1, \\ 3R(n-1) - R(n-2), & \text{for even } n > 1. \end{cases}$$

Write function `R(n)` to return the n^{th} term in Risky Sequence by recursion.

(b) Padovan Sequence $\{P(n)\}$ is the an integer sequence defined by the recurrence relation

$$P(n) = P(n-2) + P(n-3),$$

with initial values $P(0) = P(1) = P(2) = 1$. Write function `P(n)` to return the n^{th} term in Padovan Sequence by recursion.

(c) Moser-de Bruijn sequence $\{S(n)\}$ is defined by the recurrence relation

$$S(2n) = 4S(n) \text{ and } S(2n+1) = 4S(n) + 1,$$

with initial values $S(0) = 0$ and $S(1) = 1$. Write function `S(n)` to return the n^{th} term in Moser-de Brujin Sequence by recursion.

2. (a) Company A has n tasks for employees every day. Each task has a corresponding number i, ranging from 1 to n. An employee taking part in task i can earn i dollars. Each employee is initially assigned with a task k, where $1 \le k \le n$.

To prevent employees from getting too tired, after taking part in task i, there is a probability of 0.2 and 0.3 to unlock the tasks $2i$ and $2i+1$ respectively for an employee if the tasks exist. Note that the two events are independent. As the employees are enthusiastic in work, if a task is unlocked, they will take part in it.

Let users input n and k, write function `expected(x)` to calculate the expected earning for an employee in a day by recursion and calculate the answer of `expected(k)`.

Here are some examples.

If $n = 5$ and $k = 1$, $expected(1) = 1 + 0.2 \times (2 + 0.2 \times 4 + 0.3 \times 5) + 0.3 \times 3 = 2.76$

If $n = 24$ and $k = 6$, $expected(6) = 6 + 0.2 \times (12 + 0.2 \times 24) + 0.3 \times 13 = 13.26$.

If $n = 100$ and $k = 3$, $expected(3) = 17.0811$

(b) Company B also has n tasks for employees every day. Each task has a corresponding number i, ranging from 1 to n. An employee taking part in task i can earn d_i dollars. Every task i has a stamina cost c_i, meaning that taking part in task i costs an employee c_i stamina. $d_1, d_2, \dots, d_n$ are stored in vector `d` and $c_1, c_2, \dots, c_n$ are stored in vector `c`.

An employee initially has $2n$ stamina. When the stamina becomes 0, the employee cannot work. If the remaining stamina is smaller than c_i, the employee cannot take part in task i. By exhausting every combination of tasks choices using recursion, write

`maxearn(d,c,remainingstamina)` to calculate the maximum possible amount of money an employee can earn.

For example, if $n = 10$, $d = \{1, 3, 7, 6, 8, 9, 2, 4, 4, 8\}$ and $c = \{5, 1, 2, 2, 4, 1, 3, 6, 3, 7\}$, an employee can choose task 2, 3, 4, 5, 6, 9, 10 to earn 45 dollars by using 20 stamina. You may use this example to test your solution.

3. Player 1 and Player 2 are playing a game. In this game, there are n toys on the table, where $n \geq 1$ is any positive integer. This game starts with player 1 and both players take turns to take action. For each action, a player can take 1 to $\min(5, \#\text{ of remaining toys})$ toys away from the table. When a player takes the last toy away from the table, the player wins.

(a) Write R codes to let user input n with the prompt "`Enter the number of toys:`". If the user's input is not a positive integer, print "`Please enter a positive integer!`" and ask him to input again until it is correct.

(b) Write R codes to simulate the whole game.

Program Flow:

1. Input n as described in (a) and begin with player 1.
2. Ask the corresponding player to enter the number of toys to be taken away from the table with prompt "`The current number of toys on the table is i. Player j, please enter the number of toys to be taken away (between 1 to k):`", where i should be the current number of toys on the table, j should be either 1 or 2 depending on which player's turn it is, k is $min(5, i)$. When the player enters an invalid number, the program shall display a warning message: "`Invalid input!`". This phase finishes this when the player inputs correctly.
3. Repeat Step 2 until there is no toys left on the table.
4. Display the messages "`Player 1 wins!`" or "`Player 2 wins!`" accordingly.

1(a)

```
R <- function(n) {
    if (n==0 || n==1) {
        return(1)
    }
    if (n %% 2 == 1) {  #odd
        return(4*R(n-1) - 3*R(n-2))
    }
    else {  #even
        return(3*R(n-1) - R(n-2))
    }
}
```

best do n<=1

```
# Q1b
P <- function(n){
  if (n<=2) # n is an integer>=0
    return (1)
  return (P(n-2)+P(n-3))
}

# Q1c
S <- function(n){
  if (n<=0) # n is an integer>=0
    return(0)
  if (n==1)
    return(1)
  if (n%%2==0) # even n
    return(4*S(n/2))
  return(4*S(n%/%2)+1) # odd n
}
```

2(a)

```
expected <- function (k) {
    if (k > n) {
        return (0)
    }
    return (k + 0.2 * expected (2 * k) + 0.3 * expected (2 * k + 1))
}
```

(b)

```
maxearn <- function(d,c, remainingstamina) {
    num_jobs <- length(d)
    temp_earn <- d()
    for (i in 1:numjobs) {
        if ( remainingstima >= c[i] ) {
            temp_earn <- c(tmp_earn, d+ maxearn ( d[i+1:num_jobs],
                                                  c[i+1:num_jobs],
                                                  remainingstamina - c[i] )
        }
    }
    return ( max ( temp-earn) )
}
```

3(a)

```
repeat {
    n<- readline(prompt= "Enter the number of toys : \n")
    if ( class(n) != "integer" || n <=0) {
        cat ("Please enter a positive integer !" )
    }
    else {break}
}
```

yields to as.numeric

(b)

```
player ← 1

repeat {

    # read input and check
    sprintf( " The .... is %d, Player %d, ... 1 to %d \n", n, player, min(5, n))
    inp ← as.numeric( readline())

    repeat {
        if (inp < 1 or inp > min(5, n) ) {
            cat c(" Invalid input !")
        } else { break }
    # chek end
    n ← n - inp
    if ( n == 0 ) {
        sprintfc(" Player %d wins !", player )
        break
    }

# update
    if (player == 1) player ← 2 else player ← 1

}
```

```r
# Q2a
n <- as.integer(readline(prompt="Input n: "))
k <- as.integer(readline(prompt="Input k: "))
expected <- function(x){
  if (x*2>n) return(x)
  if (x*2+1>n) return(x+0.2*expected(x*2))
  return(x+0.2*expected(x*2)+0.3*expected(x*2+1))
}
expected(k)

# Q2b
maxearn <- function(d,c,remainingstamina){
  n <- length(d)
  if (n==1) # base case
    return(ifelse(remainingstamina>=c,d,0))
  if (remainingstamina>=c[n])
  # if there is enough stamina,
  # compare the money earned by the two options
  # (choosing this task or not)
    return(max(
      d[n]+maxearn(d[1:(n-1)],c[1:(n-1)],remainingstamina-c[n]),
      # pick up task n
      maxearn(d[1:(n-1)],c[1:(n-1)],remainingstamina)
      # do not choose this task
    ))
  return(maxearn(d[1:(n-1)],c[1:(n-1)],remainingstamina)) # not enough stamina
}
```

```r
# Q3a
repeat{
  n <- as.numeric(readline(prompt="Enter the number of toys: "))
  if (is.na(n)) cat("Please enter a positive integer!")
  # check characters
  else if (n<=0) cat("Please enter a positive integer!")
  # check non-positive number
  else if (n-floor(n)>0) cat("Please enter a positive integer!")
  # check floats
  else break
}

#Q3b
i <- n # i stores the current number of toys on the table
j <- 2 # j represents which player's turn
while (i>0){
  j=(2-j)%%2+1 # switch turn
  repeat{
    input <- as.numeric(readline(prompt=
    cat("The current number of toys on the table is",i,". Player",j,", please en

    # check whether the input is valid
    if (is.na(input)) cat("Invalid input! ")
    else if (input<=0|input>min(5,i)) cat("Invalid input! ")
    else if (input-floor(input)>0) cat("Invalid input! ")
    else break # break if it is valid
  }
  i=i-input # update the number of toys on the table
}
if (j==1){
  print("Player 1 wins!")
} else print("Player 2 wins!")
```

STAT2005 Introduction to Programming Languages for Statistics
Sample Midterm Examination Paper

Answer ALL questions.

Question 1 (27 marks)

(a) (7 marks) Write the R codes to create the following object named **mylist**.

```
> mylist
$a
[1] 1 2 3 4 5

$b
[1] "a" "b"

$c
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

(b) (10 marks) A survey was conducted from a series of software workshops. The information collected from the survey include

- Workshop – software introduced at the workshop
- Gender – gender of participant
- Q1 – The instructor was well prepared.
- Q2 – The instructor communicated well.
- Q3 – The course materials were helpful.
- Q4 – Overall, I found this workshop useful.

The data are stored in a data frame named **survey** as shown below.

```
> survey
  workshop gender q1 q2 q3 q4
1        R Female  4  3  4  5
2     SPSS   Male  3  4  3  4
3     <NA>   <NA>  3  2 NA  3
4     SPSS Female  5  4  5  3
5    STATA Female  4  4  3  4
6     SPSS Female  5  4  3  5
```

Write the R codes to create this data frame.

(c) (3 marks) Create a data frame consisting of only the first two columns of **survey**.

(d) (3 marks) Create a data frame consisting of only the first and last row of **survey**.

(e) (4 marks) Replace all "**Female**" by "**F**" and "**Male**" by "**M**" in **survey**.

Question 2 (19 marks)

(a) (5 marks) With the use of **sample()** function, write down a command to generate a sample from the distribution $f_X(x) = \Pr(X = x)$ given below.

x	$f_X(x)$
1	0.2
2	0.4
3	0.3
4	0.1

(b) (8 marks) Generate 2,000 random sample from $f_X(x)$ and save them as **r**. Transform **r** into a 1,000-by-2 matrix and save them again as **r**, such that each row in **r** represents a bivariate sample (x_1, x_2).

(c) (6 marks) Produce a two-way table showing the frequency count for each combination of (x_1, x_2) using the sample obtained in part (b). A sample output is shown below.

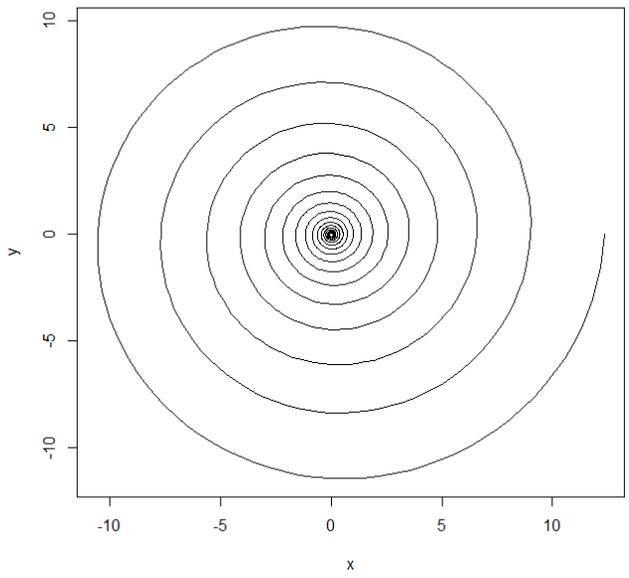
```
    1   2   3   4
1  45  84  58  26
2  76 155 114  40
3  53 124  86  32
4  22  45  29  11
```

Question 3 (18 marks)

A spiral can be described using the following equations.

$$\begin{cases} x = e^{0.05\theta} \cos\theta, \\ y = e^{0.05\theta} \sin\theta, \end{cases} \quad -16\pi \leq \theta \leq 16\pi.$$

Plot this spiral using R. A sample is shown below.



Hint: compute all the (x, y) coordinates along the given range of θ and then use **`plot()`**.

Question 4 (18 marks)

Each new term in the Fibonacci sequence is generated by adding the previous two terms. By starting with 1 and 1, the first 10 terms will be:

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

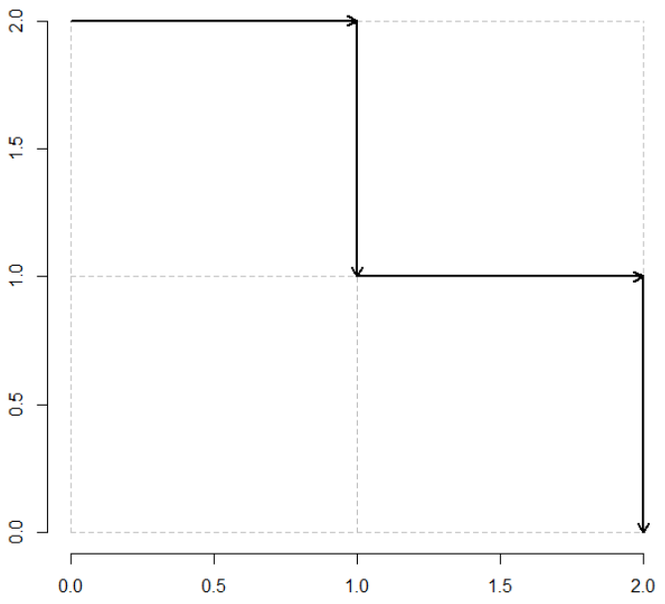
By considering the terms in the Fibonacci sequence whose values do not exceed 1,000,000,000, find the sum of the even-valued terms.

Question 5 (18 marks)

Use the following command to generate an empty plot.

```
plot(0, 0, type="n", xlim=c(0,2), ylim=c(0,2), bty="n",
xlab="", ylab="")
```

Use low level graphic functions to generate the following plot.



Note: the dashed lines are of line type 2, the arrows are of double line width and the arrow heads have length 0.1.

End of Questions

Q1(a)

```
mylist ← list ("a" = c(1,2,3,4,5), "b" = c("a", "b"),
               "c" = matrix(1:6, nrow=2))
```

(b)

```
survey ← data.frame(
    workshop = c("R", "SPSS", NA, "SPSS", "STATA", "SPSS")   ← can be factor or not
    gender = c("Female", "Male", NA, "Female", "Female", "Female")
    q1 = cc 4, 3, 3, 5, 4, 5)
    q2 = cc 3, 4, 2, 4, 4, 4)
    q3 = c(4, 3, NA, 5, 3, 3)
    q4 = c(5, 4, 3, 3, 4, 5)
)
```

(c)

```
first2 col ← survey[, c(1,2)]
```

(d)

```
first last ← survey[c(1, 6), ]
```

nrow(survey)

(e)

```
survey$gender ← factor(survey$gender)
level(survey$gender) ← c("F", "M")
```

Q2(a)

```
sample(c(1,2,3,4), size = 1, prob = c(0.2, 0.4, 0.3, 0.1))
```

replace=T..

(b)

```
r← sample(c(1,2,3,4), size = 2000, prob = c(0.2, 0.4, 0.3, 0.1), replace = T)
r← matrix(r, nrow = 1000)
```

(c)

```
table( r[,1], r[,2])
```

Q3

pi/50

```
thetas <- seq(-16*pi, 16*pi, 0.0001)
xs <- exp(0.05 * thetas) * cos(thetas)
ys <- exp(0.05 * thetas) * sin(thetas)
plot(xs, ys, type="l")
```

Q4

```
max_num <- 1 000 000 000

sum <- 0

fibs <- 1
fibl <- 1


while (fibl <= max_num ){
    if(fibl %% 2 == 0 ){
        sum <- sum+ fibl
    }

    tmp <- fibl+ fibs
    fibs <- fibl
    fibl <- tmp
}
cat (Sum)
```

Q5

```
grid( lwd=2, lty=2)

arrows( 0, 2, 1, 2, lwd = 2, head.length = 0.1)
arrows(1, 2, 1, 1, lwd = 2, head.length = 0.1)
arrows(1, 1, 2, 1, lwd = 2, head.length = 0.1)
arrows(2, 1, 2, 0, lwd = 2, head.length = 0.1)
```

```
# Q5
plot(0, 0, type="n", xlim=c(0,2), ylim=c(0,2), bty="n", xlab="", ylab="")
segments(0, 0, 2, 0, col="grey", lty=2) # "grey" is optional
segments(0, 0, 0, 2, col="grey", lty=2)
segments(0, 1, 2, 1, col="grey", lty=2)
segments(1, 0, 1, 2, col="grey", lty=2)
segments(0, 2, 2, 2, col="grey", lty=2)
segments(2, 0, 2, 2, col="grey", lty=2)
arrows(0, 2, 1, 2, lwd=2, length=0.1)
arrows(1, 2, 1, 1, lwd=2, length=0.1)
arrows(1, 1, 2, 1, lwd=2, length=0.1)
arrows(2, 1, 2, 0, lwd=2, length=0.1)
```