Labs - Practice Course

Astier Guillaume, Lefebvre Loic, Morit Luca

23/10/2026



Calculate disk usage

Role-playing dice

Sticks Game

Sticks Game

The Cash Register



Calculate disk usage



Introduction

Write a script **calcul_disk_usage.sh** which calculates the disk space occupied by a group of files of a certain type :

- The target folder (first argument).

These three parameters are passed as arguments to your script in the specified order.

If the shell script **calcul_disk_usage.sh** is not called with the right number of arguments or false arguments (ex: non-numeric value for the depth), the corresponding error is displayed and a help message for using the command is shown. You can also control :



Introduction

Write a script **calcul_disk_usage.sh** which calculates the disk space occupied by a group of files of a certain type :

- The target folder (first argument).
- The type of the searched files, with a specific extension (second argument).

These three parameters are passed as arguments to your script in the specified order.

If the shell script **calcul_disk_usage.sh** is not called with the right number of arguments or false arguments (ex: non-numeric value for the depth), the corresponding error is displayed and a help message for using the command is shown. You can also control :



Introduction

Write a script **calcul_disk_usage.sh** which calculates the disk space occupied by a group of files of a certain type :

- The target folder (first argument).
- The type of the searched files, with a specific extension (second argument).
- The maximum depth of research (third argument).

These three parameters are passed as arguments to your script in the specified order.

If the shell script **calcul_disk_usage.sh** is not called with the right number of arguments or false arguments (ex: non-numeric value for the depth), the corresponding error is displayed and a help message for using the command is shown. You can also control :



Introduction

Write a script **calcul_disk_usage.sh** which calculates the disk space occupied by a group of files of a certain type :

- The target folder (first argument).
- The type of the searched files, with a specific extension (second argument).
- The maximum depth of research (third argument).

These three parameters are passed as arguments to your script in the specified order.

If the shell script **calcul_disk_usage.sh** is not called with the right number of arguments or false arguments (ex: non-numeric value for the depth), the corresponding error is displayed and a help message for using the command is shown. You can also control :

- If the target folder exist and is a directory.



Introduction

Write a script **calcul_disk_usage.sh** which calculates the disk space occupied by a group of files of a certain type :

- The target folder (first argument).
- The type of the searched files, with a specific extension (second argument).
- The maximum depth of research (third argument).

These three parameters are passed as arguments to your script in the specified order.

If the shell script **calcul_disk_usage.sh** is not called with the right number of arguments or false arguments (ex: non-numeric value for the depth), the corresponding error is displayed and a help message for using the command is shown. You can also control :

- If the target folder exist and is a directory.
- If the result of your search contain at least a file.



Introduction

Write a script **calcul_disk_usage.sh** which calculates the disk space occupied by a group of files of a certain type :

- The target folder (first argument).
- The type of the searched files, with a specific extension (second argument).
- The maximum depth of research (third argument).

These three parameters are passed as arguments to your script in the specified order.

If the shell script **calcul_disk_usage.sh** is not called with the right number of arguments or false arguments (ex: non-numeric value for the depth), the corresponding error is displayed and a help message for using the command is shown. You can also control :

- If the target folder exist and is a directory.
- If the result of your search contain at least a file.
- That the specified extension is in a valid list that you defined.



Tips

- Make a first version without any argument, with a null depth of research or a full depth, and a specific extension with a start of research from your current folder. (ex : All files with txt extension from the current folder in any child folder)



Tips

- Make a first version without any argument, with a null depth of research or a full depth, and a specific extension with a start of research from your current folder. (ex : All files with txt extension from the current folder in any child folder)
- A second script version with the target folder argument



Tips

- Make a first version without any argument, with a null depth of research or a full depth, and a specific extension with a start of research from your current folder. (ex : All files with txt extension from the current folder in any child folder)
- A second script version with the target folder argument
- A third script version with the extension of the searched file



Tips

- Make a first version without any argument, with a null depth of research or a full depth, and a specific extension with a start of research from your current folder. (ex : All files with txt extension from the current folder in any child folder)
- A second script version with the target folder argument
- A third script version with the extension of the searched file
- A fourth script argument with the depth of your research



Help

The following concepts will be used to make the script :

- The bash command du (for Disk Usage) which return the occupied disk space for a file.



Help

The following concepts will be used to make the script :

- The bash command du (for Disk Usage) which return the occupied disk space for a file.
- The bash command find to search and print a list of files according to their extension (one filter from many possible). For this command, beware of file's name with spaces `(IFS=$'\n')`



Help

The following concepts will be used to make the script :

- The bash command du (for Disk Usage) which return the occupied disk space for a file.
- The bash command find to search and print a list of files according to their extension (one filter from many possible). For this command, beware of file's name with spaces `(IFS=$'\n')`
- And of course, the math expressions to calculate the total amount of disk space with an addition.



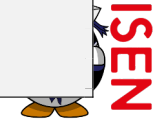
Example of execution

The display must necessarily look like this example:

```
username@hostname:$ ./calcul_disk_usage.sh $HOME txt 2
/home/catanese/phpmyadmin.txt size = 4
/home/catanese/.minetest/debug.txt size = 456
/home/catanese/Vidéos/torrent.txt size = 4
/home/catanese/prive/ssh_one_and_one.txt size = 4
/home/catanese/install/torrent.txt size = 4
/home/catanese/bin/README_yGenerate_QCM.txt size = 4
/home/catanese/bin/monfichier.txt size = 4
Number of files found 7, size : 480 octets
```

```
username@hostname:$ ./calcul_disk_usage.sh $HOME txt invalid
Search depth must be an integer
Command syntax ./calcul_disk_usage.sh TARGET_DIRECTORY EXTENSION DEPTH
```

```
username@hostname:$ ./calcul_disk_usage.sh invalid txt 2
The name of the directory entered does not exist
Command syntax ./calcul_disk_usage.sh TARGET_DIRECTORY EXTENSION DEPTH
```



```
username@hostname:$ ./calcul_disk_usage.sh $HOME invalid 2
Extension name is not in the list
Command syntax ./calcul_disk_usage.sh TARGET_DIRECTORY EXTENSION DEPTH
```

```
username@hostname:$ ./calcul_disk_usage.sh $HOME/workspace java 2
Number of files found 0, size : 0 octets
```



```
username@hostname:$ ./calcul_disk_usage.sh $HOME/workspace java 4
/home/catanese/workspace/VISUEL/src/geodesie/CRepereMadone.java size = 4
/home/catanese/workspace/VISUEL/src/geodesie/CPoint3D.java size = 20
/home/catanese/workspace/VISUEL/src/enregistrement/CEnregistrementVG0XThread.java
   size = 4
/home/catanese/workspace/VISUEL/src/enregistrement/CEnregistrementKPCEThread.java
   size = 4…

/home/catanese/workspace/VISUEL/src/ihm/FenRejeu.java size = 20
/home/catanese/workspace/VISUEL/src/ihm/FenPrepa.java size = 16
/home/catanese/workspace/VISUEL/src/ihm/FenObj.java size = 116
/home/catanese/workspace/VISUEL/src/ihm/FullScreenToggleAction.java size = 4
/home/catanese/workspace/Test_Auto_IHM_Java/src/simple_ihm/Main_IHM.java size = 8
Number of files found 58, size : 1336 octets
```



Role-playing dice



Introduction



The goal of this exercise is to create a script that will display X dice with a random number between 1 and 6 in the terminal.



Display

The dice have to be like that :

```
   .-------.
  /   *   /|
 /_______/ |
 |       |*|
 |   *   | /
 |       |/
 '-------'
```



The display must necessarily look like this example :

```
isen@isen $ ./my-dice
how many dice : 3


       .-------.
      /   1   /|
     /_______/ |
     |       |1|
     |   1   | /
     |       |/
     '-------'
       .-------.
      /   6   /|
     /_______/ |
     |       |6|
     |   6   | /
     |       |/
     '-------'
       .-------.
      /   4   /|
     /_______/ |
     |       |4|
     |   4   | /
```



Requirement

- The sample of the ASCII art of the dice is present on your docker instance : /opt/dice.ascii



Requirement

- The sample of the ASCII art of the dice is present on your docker instance : /opt/dice.ascii
- The script must have contextual help with the '-h' option.



Requirement

- The sample of the ASCII art of the dice is present on your docker instance : /opt/dice.ascii
- The script must have contextual help with the '-h' option.
- The management of the random must be done with the command *shuf* (man shuf).



Requirement

- The sample of the ASCII art of the dice is present on your docker instance : /opt/dice.ascii
- The script must have contextual help with the '-h' option.
- The management of the random must be done with the command *shuf* (man shuf).
- The script name is : my-dice and upload on Moodle.



Requirement

- The sample of the ASCII art of the dice is present on your docker instance : /opt/dice.ascii
- The script must have contextual help with the '-h' option.
- The management of the random must be done with the command *shuf* (man shuf).
- The script name is : my-dice and upload on Moodle.
- The script can be executed without option with user interaction (like the example of "Display").



Requirement

- The sample of the ASCII art of the dice is present on your docker instance : /opt/dice.ascii
- The script must have contextual help with the '-h' option.
- The management of the random must be done with the command *shuf* (man shuf).
- The script name is : my-dice and upload on Moodle.
- The script can be executed without option with user interaction (like the example of "Display").
- The script can be executed with the option '-n' followed by a number (ex: ./my-dice -n 3).



Requirement

- The sample of the ASCII art of the dice is present on your docker instance : /opt/dice.ascii
- The script must have contextual help with the '-h' option.
- The management of the random must be done with the command *shuf* (man shuf).
- The script name is : my-dice and upload on Moodle.
- The script can be executed without option with user interaction (like the example of "Display").
- The script can be executed with the option '-n' followed by a number (ex: ./my-dice -n 3).
- The minimum number of dice must be 1 and the maximum must be 6. You need to check it and return the contextual help if it's not good.



Advise

- You can use sed to replace something in the dice.



Advise

- You can use sed to replace something in the dice.
- You can create a variable which will sotck the return of the shuf command for each die.



Advise

- You can use sed to replace something in the dice.
- You can create a variable which will sotck the return of the shuf command for each die.
- Pay attention to the spaces of the ascii art in relation to the variable.



Sticks Game



Introduction

The goal of this exerice is to create the Sticks Game.

This game is a duel between computer and human player.

There are N sticks. Each gamer has to take 1, 2 or 3 sticks. If the gamer take the last sitck, he loses.



ISEN

Requirements

- The numbers of sticks is given in **argument**.



Requirements

- The numbers of sticks is given in **argument**.
- The numbers of sticks have to be a number between 10 and 30.



Requirements

- The numbers of sticks is given in **argument**.
- The numbers of sticks have to be a number between 10 and 30.
- The humain player is the first gamer.



Requirements

- The numbers of sticks is given in **argument**.
- The numbers of sticks have to be a number between 10 and 30.
- The humain player is the first gamer.
- When the computer play, it take randomly sitcks, but it wants to win. So if there are 3 sitcks, it take 2. If there are 2 sticks, it takes 1.



Requirements

- The numbers of sticks is given in **argument**.
- The numbers of sticks have to be a number between 10 and 30.
- The humain player is the first gamer.
- When the computer play, it take randomly sitcks, but it wants to win. So if there are 3 sitcks, it take 2. If there are 2 sticks, it takes 1.
- The random number must be between 1 and 3. The modulo can take the value of 0. In this case, a new random number must be recalculated until it has a value between 1 and 3.



Requirements

- The numbers of sticks is given in **argument**.
- The numbers of sticks have to be a number between 10 and 30.
- The humain player is the first gamer.
- When the computer play, it take randomly sitcks, but it wants to win. So if there are 3 sitcks, it take 2. If there are 2 sticks, it takes 1.
- The random number must be between 1 and 3. The modulo can take the value of 0. In this case, a new random number must be recalculated until it has a value between 1 and 3.
- All parameters must be tested. (if the script asks a number, so the script must check if the gamer typed a valid number).



Requirements

- The numbers of sticks is given in **argument**.
- The numbers of sticks have to be a number between 10 and 30.
- The humain player is the first gamer.
- When the computer play, it take randomly sitcks, but it wants to win. So if there are 3 sitcks, it take 2. If there are 2 sticks, it takes 1.
- The random number must be between 1 and 3. The modulo can take the value of 0. In this case, a new random number must be recalculated until it has a value between 1 and 3.
- All parameters must be tested. (if the script asks a number, so the script must check if the gamer typed a valid number).
- The display must follow the example.



Requirements

- The numbers of sticks is given in **argument**.
- The numbers of sticks have to be a number between 10 and 30.
- The humain player is the first gamer.
- When the computer play, it take randomly sitcks, but it wants to win. So if there are 3 sitcks, it take 2. If there are 2 sticks, it takes 1.
- The random number must be between 1 and 3. The modulo can take the value of 0. In this case, a new random number must be recalculated until it has a value between 1 and 3.
- All parameters must be tested. (if the script asks a number, so the script must check if the gamer typed a valid number).
- The display must follow the example.
- There must be a function to display the number remaining sticks .



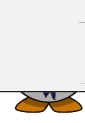
Requirements

- The numbers of sticks is given in **argument**.
- The numbers of sticks have to be a number between 10 and 30.
- The humain player is the first gamer.
- When the computer play, it take randomly sitcks, but it wants to win. So if there are 3 sitcks, it take 2. If there are 2 sticks, it takes 1.
- The random number must be between 1 and 3. The modulo can take the value of 0. In this case, a new random number must be recalculated until it has a value between 1 and 3.
- All parameters must be tested. (if the script asks a number, so the script must check if the gamer typed a valid number).
- The display must follow the example.
- There must be a function to display the number remaining sticks .
- The name of script must be : sticks_game.sh



Sample with human player is winner

```
isen@isen $ ./sticks_game.sh 10
Start Sticks Game with 10 sticks
|   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |
How many sticks do you take ? 3
|   |   |   |   |   |   |
|   |   |   |   |   |   |
Computer takes 2
|   |   |   |   |
|   |   |   |   |
How many sticks do you take ? 1
|   |   |   |
|   |   |   |
Computer takes 2
|   |
|   |
How many sticks do you take ? 1
|
|
You win
```



Sample with computer is winner

```
isen@isen $ ./sticks_game.sh 10
Start Sticks Game with 10 sticks
|   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |
How many sticks do you take ? 3
|   |   |   |   |   |   |
|   |   |   |   |   |   |
Computer takes 1
|   |   |   |   |   |
|   |   |   |   |   |
How many sticks do you take ? 3
|   |   |
|   |   |
Computer takes 2
|
|
How many sticks do you take ? 1
You lose
```



Sample with checks

```
isen@isen $ ./sticks_game.sh notNumber
The sitck number is not a number
isen@isen $ echo $?
1
isen@isen $ ./sticks_game.sh 9
The sitck number is not between 20 and 30
isen@isen $ echo $?
2
isen@isen $ ./sticks_game.sh 31
The sitck number is not between 20 and 30
isen@isen $ ./sticks_game.sh 10
Start Sticks Game with 10 sticks
|   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |
How many sticks do you take ? notNumber
It is not a number
How many sticks do you take ? 5
You have take sitck between 1 and 3
How many sticks do you take ?
```



Sticks Game



Introduction

The goal of this exerice is to create the Sticks Game.

This game is a duel between computer and human player.

There are N sticks. Each gamer has to take 1, 2 or 3 sticks. If the gamer take the last sitck, he loses.



ISEN

Requirements

- The numbers of sticks is given in **argument**.



Requirements

- The numbers of sticks is given in **argument**.
- The numbers of sticks have to be a number between 10 and 30.



Requirements

- The numbers of sticks is given in **argument**.
- The numbers of sticks have to be a number between 10 and 30.
- The humain player is the first gamer.



Requirements

- The numbers of sticks is given in **argument**.
- The numbers of sticks have to be a number between 10 and 30.
- The humain player is the first gamer.
- When the computer play, it take randomly sitcks, but it wants to win. So if there are 3 sitcks, it take 2. If there are 2 sticks, it takes 1.



Requirements

- The numbers of sticks is given in **argument**.
- The numbers of sticks have to be a number between 10 and 30.
- The humain player is the first gamer.
- When the computer play, it take randomly sitcks, but it wants to win. So if there are 3 sitcks, it take 2. If there are 2 sticks, it takes 1.
- The random number must be between 1 and 3. The modulo can take the value of 0. In this case, a new random number must be recalculated until it has a value between 1 and 3.



Requirements

- The numbers of sticks is given in **argument**.
- The numbers of sticks have to be a number between 10 and 30.
- The humain player is the first gamer.
- When the computer play, it take randomly sitcks, but it wants to win. So if there are 3 sitcks, it take 2. If there are 2 sticks, it takes 1.
- The random number must be between 1 and 3. The modulo can take the value of 0. In this case, a new random number must be recalculated until it has a value between 1 and 3.
- All parameters must be tested. (if the script asks a number, so the script must check if the gamer typed a valid number).



Requirements

- The numbers of sticks is given in **argument**.
- The numbers of sticks have to be a number between 10 and 30.
- The humain player is the first gamer.
- When the computer play, it take randomly sitcks, but it wants to win. So if there are 3 sitcks, it take 2. If there are 2 sticks, it takes 1.
- The random number must be between 1 and 3. The modulo can take the value of 0. In this case, a new random number must be recalculated until it has a value between 1 and 3.
- All parameters must be tested. (if the script asks a number, so the script must check if the gamer typed a valid number).
- The display must follow the example.



Requirements

- The numbers of sticks is given in **argument**.
- The numbers of sticks have to be a number between 10 and 30.
- The humain player is the first gamer.
- When the computer play, it take randomly sitcks, but it wants to win. So if there are 3 sitcks, it take 2. If there are 2 sticks, it takes 1.
- The random number must be between 1 and 3. The modulo can take the value of 0. In this case, a new random number must be recalculated until it has a value between 1 and 3.
- All parameters must be tested. (if the script asks a number, so the script must check if the gamer typed a valid number).
- The display must follow the example.
- There must be a function to display the number remaining sticks .



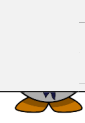
Requirements

- The numbers of sticks is given in **argument**.
- The numbers of sticks have to be a number between 10 and 30.
- The humain player is the first gamer.
- When the computer play, it take randomly sitcks, but it wants to win. So if there are 3 sitcks, it take 2. If there are 2 sticks, it takes 1.
- The random number must be between 1 and 3. The modulo can take the value of 0. In this case, a new random number must be recalculated until it has a value between 1 and 3.
- All parameters must be tested. (if the script asks a number, so the script must check if the gamer typed a valid number).
- The display must follow the example.
- There must be a function to display the number remaining sticks .
- The name of script must be : sticks_game.sh



Sample with human player is winner

```
isen@isen $ ./sticks_game.sh 10
Start Sticks Game with 10 sticks
|   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |
How many sticks do you take ? 3
|   |   |   |   |   |   |
|   |   |   |   |   |   |
Computer takes 2
|   |   |   |   |
|   |   |   |   |
How many sticks do you take ? 1
|   |   |   |
|   |   |   |
Computer takes 2
|   |
|   |
How many sticks do you take ? 1
|
|
You win
```



Sample with computer is winner

```
isen@isen $ ./sticks_game.sh 10
Start Sticks Game with 10 sticks
|   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |
How many sticks do you take ? 3
|   |   |   |   |   |   |
|   |   |   |   |   |   |
Computer takes 1
|   |   |   |   |   |
|   |   |   |   |   |
How many sticks do you take ? 3
|   |   |
|   |   |
Computer takes 2
|
|
How many sticks do you take ? 1
You lose
```



Sample with checks

```
isen@isen $ ./sticks_game.sh notNumber
The sitck number is not a number
isen@isen $ echo $?
1
isen@isen $ ./sticks_game.sh 9
The sitck number is not between 20 and 30
isen@isen $ echo $?
2
isen@isen $ ./sticks_game.sh 31
The sitck number is not between 20 and 30
isen@isen $ ./sticks_game.sh 10
Start Sticks Game with 10 sticks
|   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |
How many sticks do you take ? notNumber
It is not a number
How many sticks do you take ? 5
You have take sitck between 1 and 3
How many sticks do you take ?
```



The Cash Register



The Cash Register

The purpose of the exercise is to create a cash register for a store.



Main requirements

- The tree must be:



Main requirements

- The tree must be:
  - `/home/isen/EXAM/CashRegister/CashRegister.sh` : your script



Main requirements

- The tree must be:
  - `/home/isen/EXAM/CashRegister/CashRegister.sh` : your script
  - `/home/isen/EXAM/CashRegister/CashRegister.d` : your folder where all sales will be stored.



Main requirements

- The tree must be:
  - `/home/isen/EXAM/CashRegister/CashRegister.sh` : your script
  - `/home/isen/EXAM/CashRegister/CashRegister.d` : your folder where all sales will be stored.
  - `/home/isen/EXAM/CashRegister/CashRegister.d/YYYYMMDD.csv` : One day sales (example : 20221013.csv)



Main requirements

- The tree must be:
  - `/home/isen/EXAM/CashRegister/CashRegister.sh` : your script
  - `/home/isen/EXAM/CashRegister/CashRegister.d` : your folder where all sales will be stored.
  - `/home/isen/EXAM/CashRegister/CashRegister.d/YYYYMMDD.csv` : One day sales (example : 20221013.csv)
- One day sales are stored in CSV format like this:

  `HH-MM;customer name;amount;payment method` (example : 13-28;LEFEBVRE;28;CARD)



Main requirements

- The tree must be:
  - `/home/isen/EXAM/CashRegister/CashRegister.sh` : your script
  - `/home/isen/EXAM/CashRegister/CashRegister.d` : your folder where all sales will be stored.
  - `/home/isen/EXAM/CashRegister/CashRegister.d/YYYYMMDD.csv` : One day sales (example : 20221013.csv)
- One day sales are stored in CSV format like this:

  `HH-MM;customer name;amount;payment method` (example : 13-28;LEFEBVRE;28;CARD)
- Amounts are always whole numbers (integer)



Main requirements

- The tree must be:
  - `/home/isen/EXAM/CashRegister/CashRegister.sh` : your script
  - `/home/isen/EXAM/CashRegister/CashRegister.d` : your folder where all sales will be stored.
  - `/home/isen/EXAM/CashRegister/CashRegister.d/YYYYMMDD.csv` : One day sales (example : 20221013.csv)
- One day sales are stored in CSV format like this:

  `HH-MM;customer name;amount;payment method` (example : 13-28;LEFEBVRE;28;CARD)
- Amounts are always whole numbers (integer)
- The means of payment are only: CASH, CARD or BANK_CHECK



Step 1 : the launch menu

When you run your script, it should display the following menu:



```
isen@lefebvre_l_client ~/EXAM/CashRegister $ ./CashRegister.sh
Select your action
        add a sale: ADD
        close day: CLOSE
        quit: QUIT
nokeywork
        Please select ADD|CLOSE|QUIT
Select your action
        add a sale: ADD
        close day: CLOSE
        quit: QUIT
ADD
        add sale
Select your action
        add a sale: ADD
        close day: CLOSE
        quit: QUIT
CLOSE
        close
Select your action
        add a sale: ADD
        close day: CLOSE
        quit: QUIT
QUIT
        Good Bye
```



You must use : - a case/esac - a loop: while - the read command - 1 function per keyword, i.e. 4 functions - 1 function to display the menu



Step 2 : add a sale

You must complete your ADD function to save a sale in the file (be careful at the very first execution, it may be necessary to create some folders, file).

If ever the file of the day exists, it must be completed.

You must check when entering whether: - customer name is not empty - the amount is indeed an int - the payment method is correct

You must of course respect the example below



```
isen@lefebvre_l_client ~/EXAM/CashRegister $ ./CashRegister.sh
Select your action
        add a sale: ADD
        close day: CLOSE
        quit: QUIT
ADD
Customer:
Customer: Loic
Amount: pas un chiffre
Amount: 12
Means of payment (CARD|CASH|BANK_CHECK): ERROR
Means of payment (CARD|CASH|BANK_CHECK): CARD
Select your action
        add a sale: ADD
        close day: CLOSE
        quit: QUIT
ADD
Customer: Guillaume
Amount: 12
Means of payment (CARD|CASH|BANK_CHECK): CASH
Select your action
        add a sale: ADD
        close day: CLOSE
        quit: QUIT
QUIT
```



Step 3 : the closing of the day

You must complete your CLOSE function to display : - show total by payment method - display the total of the day



```
isen@lefebvre_l_client ~/EXAM/CashRegister $ cat CashRegister.d/20221013.csv
20-16;Loic;12;CARD
20-16;Loic;12;CARD
20-16;Loic;12;CARD
20-16;Loic;12;CARD
20-16;Loic;12;CARD
20-16;Loic;12;CARD
20-16;Guillaume;12;CASH
20-16;Guillaume;12;CASH
20-16;Guillaume;12;CASH
20-16;Guillaume;12;CASH
20-16;Guillaume;12;CASH
isen@lefebvre_l_client ~/EXAM/CashRegister $ ./CashRegister.sh
Select your action
        add a sale: ADD
        close day: CLOSE
        quit: QUIT
CLOSE
TOTAL CASH : 60
TOTAL CARD : 72
TOTAL BANK_CHECK : 0
TOTAL : 132
Select your action
        add a sale: ADD
        close day: CLOSE
```

