21/03/2023 **S. Dhanush kumar**

192121154

R programming

Assignment 2

1. The built-in vector LETTERS contains the uppercase letters of the alphabet. Produce a vector of

(i) the first 12 letters;

(ii) the odd 'numbered' letters;

(iii) the (English) consonants.

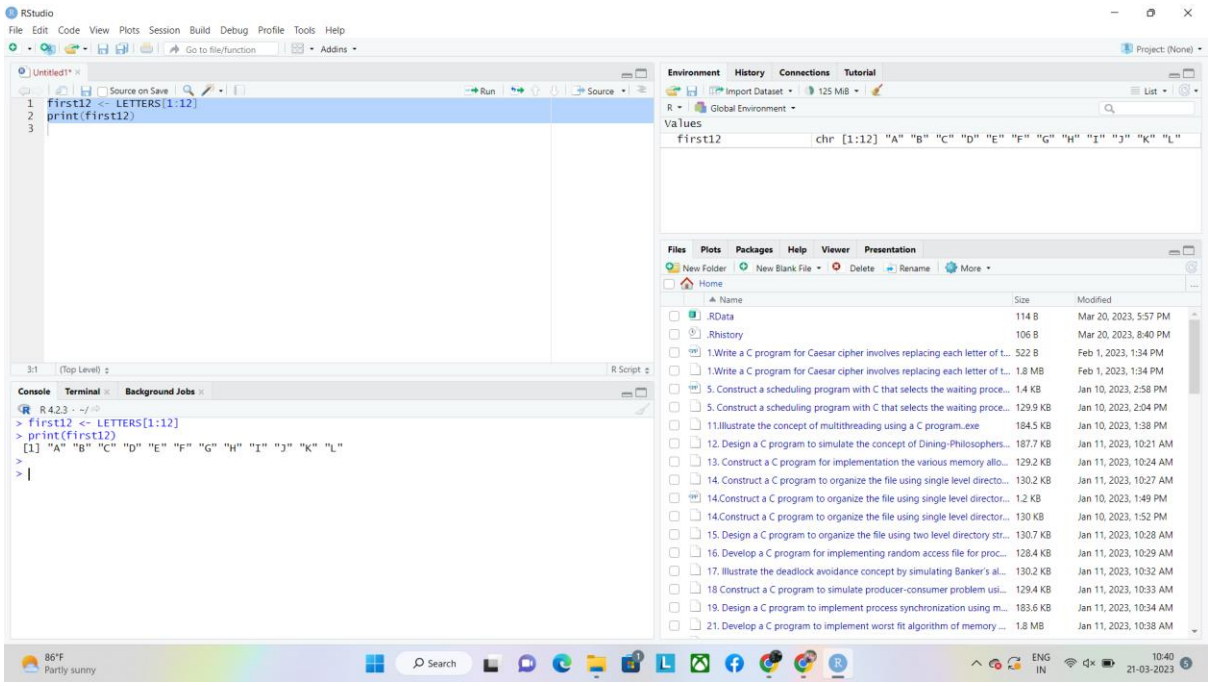
Program:

(i) the first 12 letters;

```
first12 <- LETTERS[1:12]
print(first12)
```

output:

```
> first12 <- LETTERS[1:12]
> print(first12)
 [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L"
```



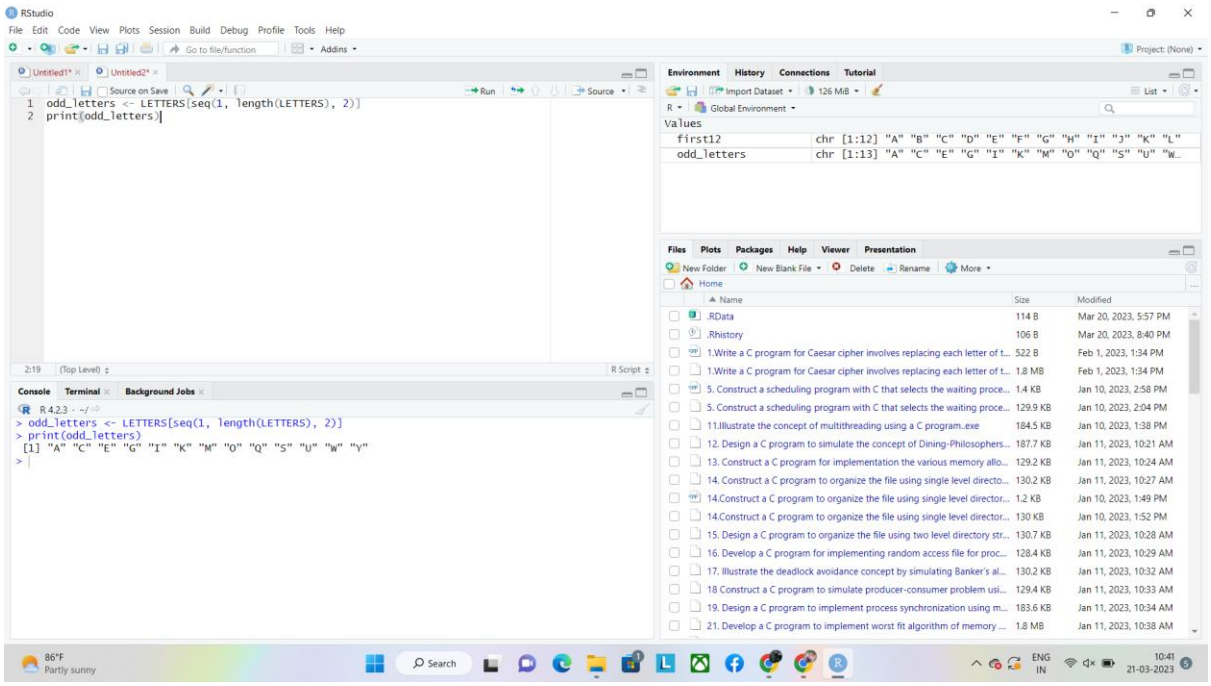
(ii) the odd 'numbered' letters;

Program:

```
odd_letters <- LETTERS[seq(1, length(LETTERS), 2)]
print(odd_letters)
```

Output:

```
  odd_letters <- LETTERS[seq(1, length(LETTERS), 2)]
> print(odd_letters)
 [1] "A" "C" "E" "G" "I" "K" "M" "O" "Q" "S" "U" "W" "Y"
```



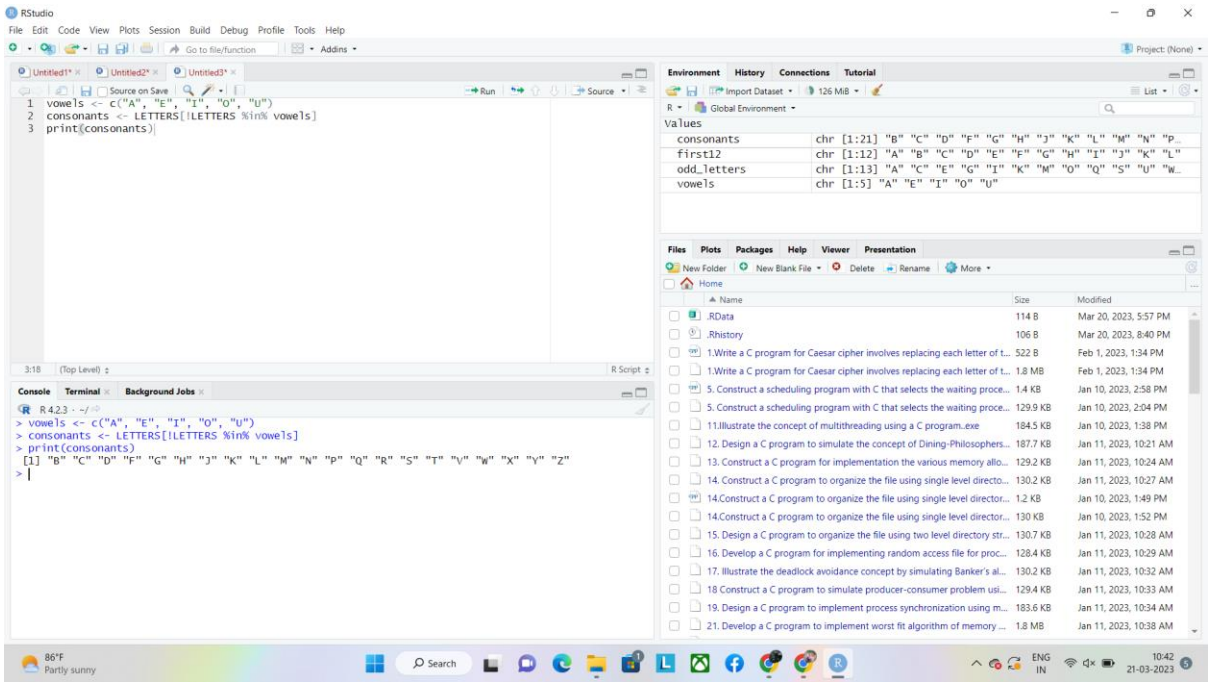
(iii) the (English) consonants.

Program:

```
vowels <- c("A", "E", "I", "O", "U")
consonants <- LETTERS[!LETTERS %in% vowels]
print(consonants)
```

output:

```
> vowels <- c("A", "E", "I", "O", "U")
> consonants <- LETTERS[!LETTERS %in% vowels]
> print(consonants)
 [1] "B" "C" "D" "F" "G" "H" "J" "K" "L" "M" "N" "P" "Q" "R" "S" "T" "V" "W" "X" "Y" "
>
```



2. The function rnorm() generates normal random variables. For instance, rnorm(10) gives a vector

of 10 i.i.d. standard normals. Generate 20 standard normals, and store them as x. Then obtain

subvectors of

(i) the entries in x which are less than 1;

(ii) the entries between – 0.5 and 1;

(iii) the entries whose absolute value is larger than 1.5.

Program:

(i) the entries in x which are less than 1;

```
# set seed for reproducibility
set.seed(123)
```

 # generate 20 standard normals

x <- rnorm(20)

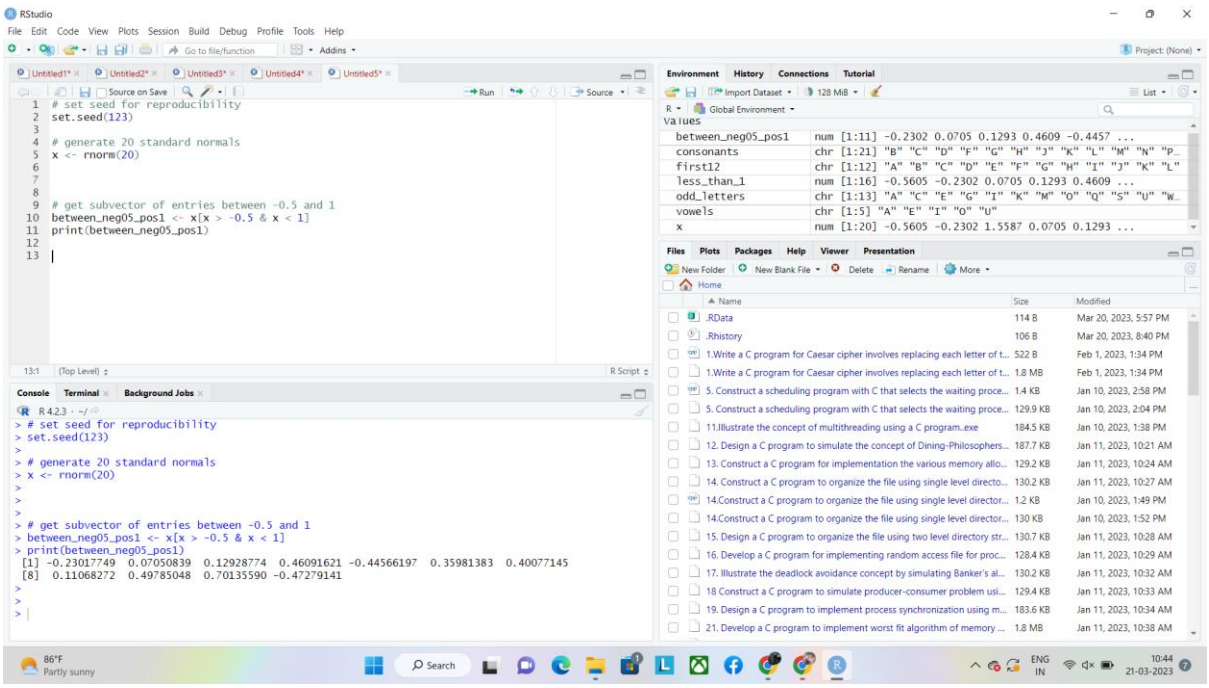
 # get subvector of entries in x which are less than 1

less_than_1 <- x[x < 1]

print(less_than_1).

Output:

```
> # set seed for reproducibility
> set.seed(123)
>
> # generate 20 standard normals
> x <- rnorm(20)
>
> # get subvector of entries in x which are less than 1
> less_than_1 <- x[x < 1]
> print(less_than_1)
 [1] -0.56047565 -0.23017749  0.07050839  0.12928774  0.46091621 -1.26506123 -0.686852
 [8] -0.44566197  0.35981383  0.40077145  0.11068272 -0.55584113  0.49785048 -1.966617
[15]  0.70135590 -0.47279141

>
```



Program:

(ii) the entries between – 0.5 and 1;

 # set seed for reproducibility

set.seed(123)

 # generate 20 standard normals

x <- rnorm(20)

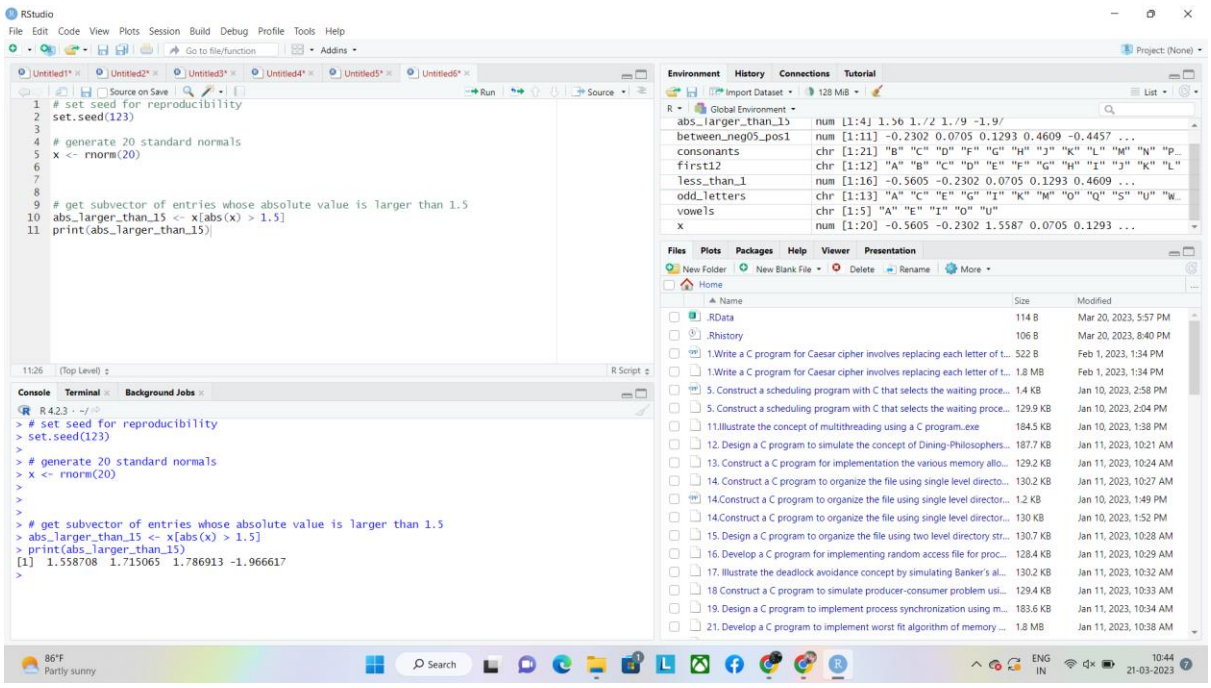
 # get subvector of entries between -0.5 and 1

between_neg05_pos1 <- x[x > -0.5 & x < 1]

print(between_neg05_pos1)

output:

```
> # set seed for reproducibility
> set.seed(123)
>
> # generate 20 standard normals
> x <- rnorm(20)
>
>
>
> # get subvector of entries between -0.5 and 1
> between_neg05_pos1 <- x[x > -0.5 & x < 1]
> print(between_neg05_pos1)
 [1] -0.23017749  0.07050839  0.12928774  0.46091621 -0.44566197  0.359813
83  0.40077145
 [8]  0.11068272  0.49785048  0.70135590 -0.47279141
>
```



Program:

(iii) the entries whose absolute value is larger than 1.5.

 # set seed for reproducibility

set.seed(123)

 # generate 20 standard normals

x <- rnorm(20)

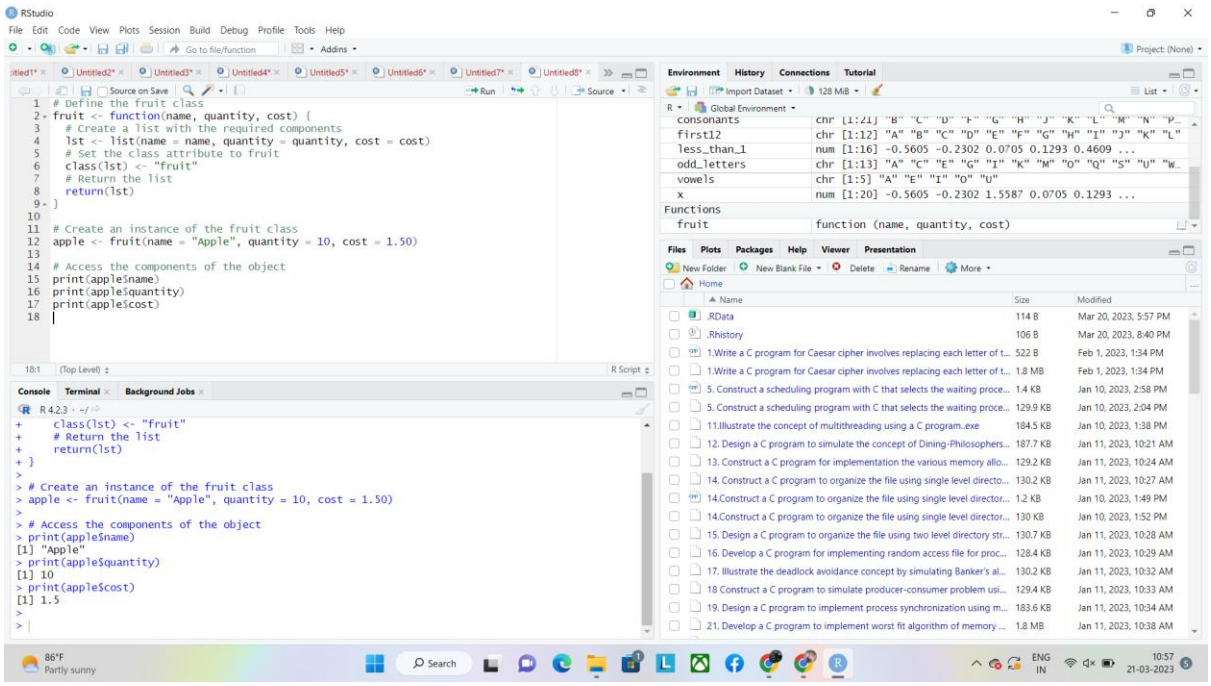
 # get subvector of entries whose absolute value is larger than 1.5

abs_larger_than_15 <- x[abs(x) > 1.5]

print(abs_larger_than_15)

output:

```
> # set seed for reproducibility
> set.seed(123)
>
> # generate 20 standard normals
> x <- rnorm(20)
>
>
>
> # get subvector of entries whose absolute value is larger than 1.5
> abs_larger_than_15 <- x[abs(x) > 1.5]
> print(abs_larger_than_15)
[1]  1.558708  1.715065  1.786913 -1.966617

>
```



3. Solve the following system of simultaneous equations using matrix methods.

a + 2b + 3c + 4d + 5e = −5

2a + 3b + 4c + 5d + e = 2

3a + 4b + 5c + d + 2e = 5

4a + 5b + c + 2d + 3e = 10

5a + b + 2c + 3d + 4e = 11

Program:

```
# Define the matrix A and vector b
A <- matrix(c(1, 2, 3, 4, 5,
              2, 3, 4, 5, 1,
              3, 4, 5, 1, 2,
              4, 5, 1, 2, 3,
              5, 1, 2, 3, 4), nrow = 5, byrow = TRUE)
b <- c(-5, 2, 5, 10, 11)
```

 # Solve the system using the solve function

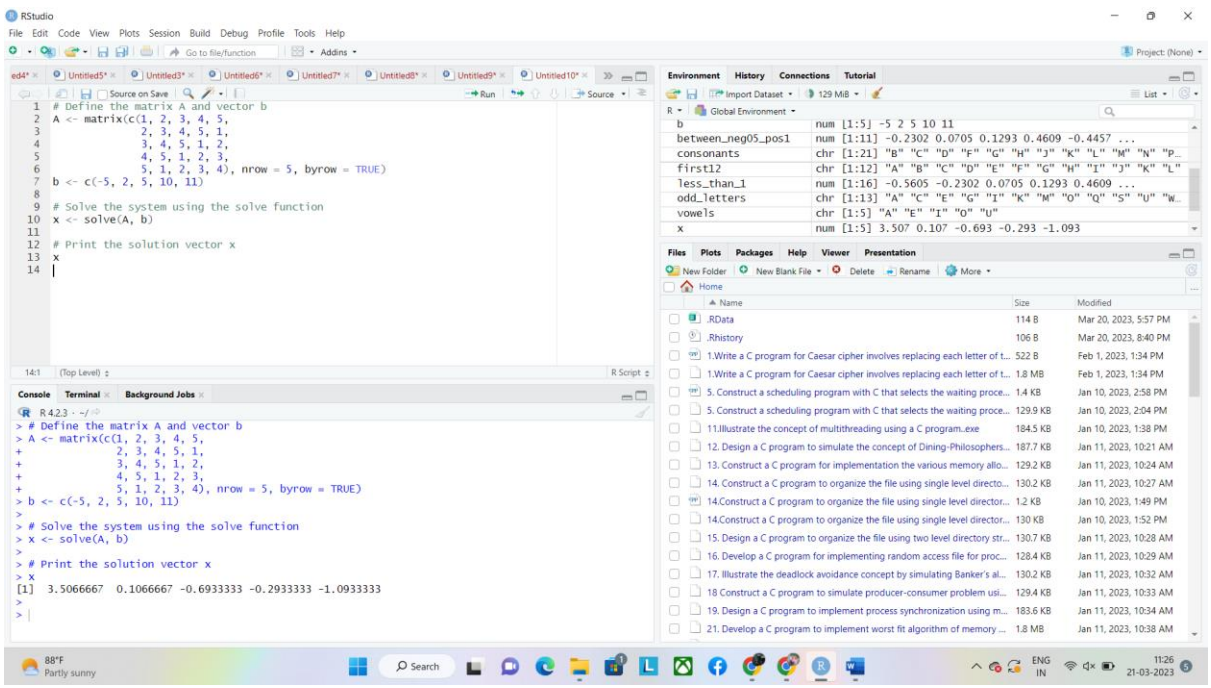
x <- solve(A, b)

 # Print the solution vector x

X

Output:

```
> # Define the matrix A and vector b
> A <- matrix(c(1, 2, 3, 4, 5,
+               2, 3, 4, 5, 1,
+               3, 4, 5, 1, 2,
+               4, 5, 1, 2, 3,
+               5, 1, 2, 3, 4), nrow = 5, byrow = TRUE)
> b <- c(-5, 2, 5, 10, 11)
>
> # Solve the system using the solve function
> x <- solve(A, b)
>
> # Print the solution vector x
> x
[1]  3.5066667  0.1066667 -0.6933333 -0.2933333 -1.0933333
>
```



4. Create a factor object for an apple color such as 'green', 'green', 'yellow', 'red', 'red', 'red','

green'. Print the factor and applying the nlevels function to know the number of distinct

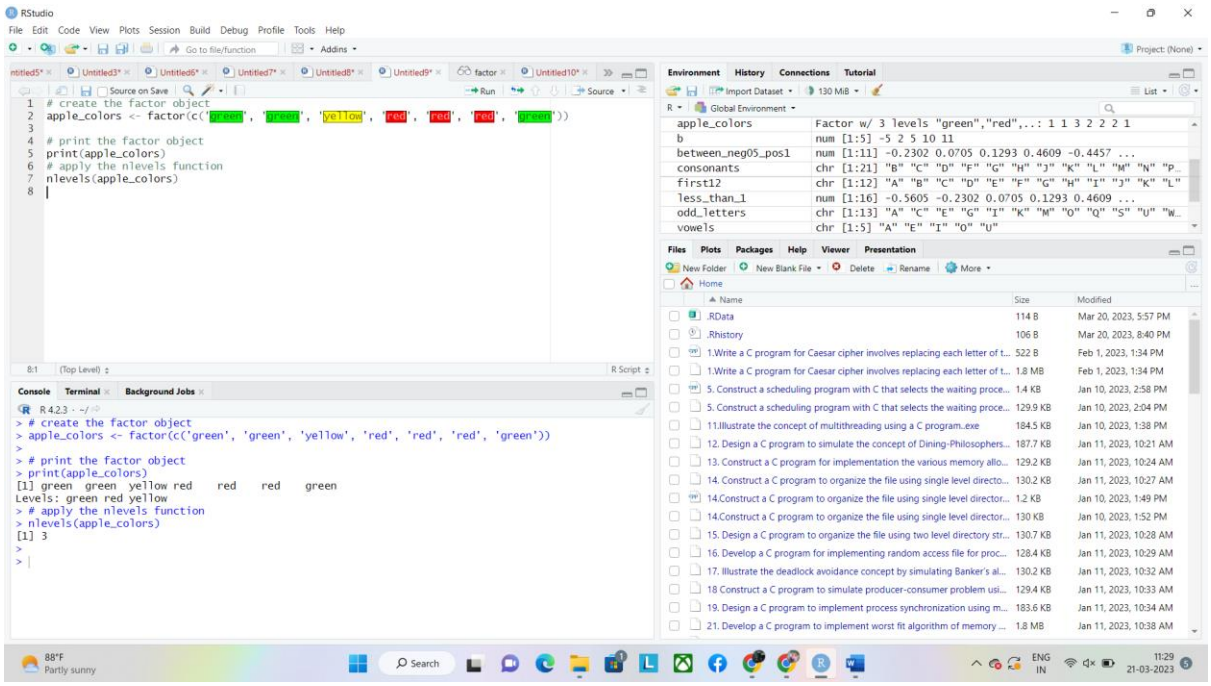
values

program:

```
# create the factor object

apple_colors <- factor(c('green', 'green', 'yellow', 'red', 'red', 'red', 'green'))


# print the factor object

print(apple_colors)

# apply the nlevels function

nlevels(apple_colors)
```

output:

```
> # create the factor object
> apple_colors <- factor(c('green', 'green', 'yellow', 'red', 'red', 'red', 'green'))
>
> # print the factor object
> print(apple_colors)
[1] green  green  yellow red    red    red    green
Levels: green red yellow
> # apply the nlevels function
> nlevels(apple_colors)
[1] 3
>

>
```



5. Create an S3 object of class fruit contains a list with following required components such

as name, quantity, cost and also Define and create s4 objects.Define a reference class of

fruit

program:

```
# Define the fruit class
fruit <- function(name, quantity, cost) {
  # Create a list with the required components
  lst <- list(name = name, quantity = quantity, cost = cost)
  # Set the class attribute to fruit
  class(lst) <- "fruit"
  # Return the list
  return(lst)
}
```

 # Create an instance of the fruit class

apple <- fruit(name = "Apple", quantity = 10, cost = 1.50)

 # Access the components of the object

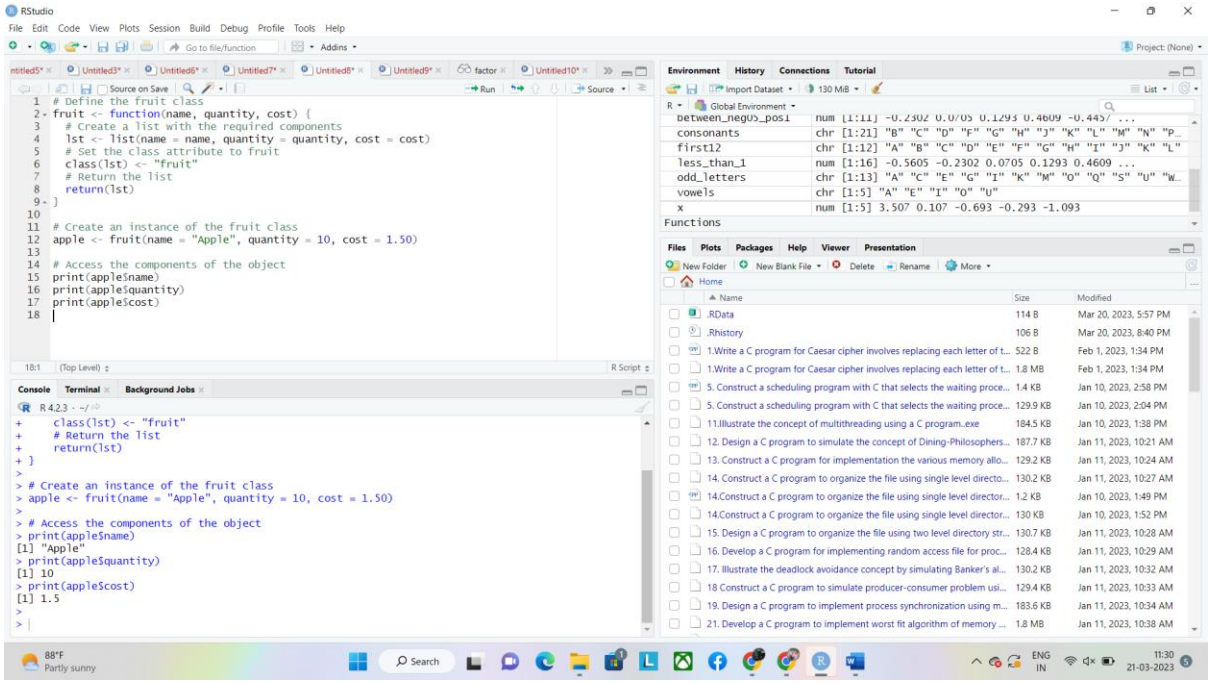
print(apple$name)

print(apple$quantity)

print(apple$cost)

output:

```
> # Define the fruit class
> fruit <- function(name, quantity, cost) {
+     # Create a list with the required components
+     lst <- list(name = name, quantity = quantity, cost = cost)
+     # Set the class attribute to fruit
+     class(lst) <- "fruit"
+     # Return the list
+     return(lst)
+ }
>
> # Create an instance of the fruit class
> apple <- fruit(name = "Apple", quantity = 10, cost = 1.50)
>
> # Access the components of the object
> print(apple$name)
[1] "Apple"
> print(apple$quantity)
[1] 10
> print(apple$cost)
[1] 1.5
>

>
```


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Untitled8*
Source on Save
Run
Source
1 # Define the fruit class
2 fruit <- function(name, quantity, cost) {
3 # Create a list with the required components
4 lst <- list(name = name, quantity = quantity, cost = cost)
5 # Set the class attribute to fruit
6 class(lst) <- "fruit"
7 # Return the list
8 return(lst)
9 }
10
11 # Create an instance of the fruit class
12 apple <- fruit(name = "Apple", quantity = 10, cost = 1.50)
13
14 # Access the components of the object
15 print(apple$name)
16 print(apple$quantity)
17 print(apple$cost)
18
18:1 (Top Level)
R Script
Console Terminal Background Jobs
R 4.2.3 · ~/
+ class(lst) <- "fruit"
+ # Return the list
+ return(lst)
+ }
>
> # Create an instance of the fruit class
> apple <- fruit(name = "Apple", quantity = 10, cost = 1.50)
>
> # Access the components of the object
> print(apple$name)
[1] "Apple"
> print(apple$quantity)
[1] 10
> print(apple$cost)
[1] 1.5
>
Environment History Connections Tutorial
Import Dataset
130 MiB
List
R
Global Environment
between_neg05_pos1 num [1:11] -0.2302 0.0705 0.1293 0.4609 -0.4457 ...
consonants chr [1:21] "B" "C" "D" "F" "G" "H" "J" "K" "L" "M" "N" "P"...
first12 chr [1:12] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L"
less_than_1 num [1:16] -0.5605 -0.2302 0.0705 0.1293 0.4609 ...
odd_letters chr [1:13] "A" "C" "E" "G" "I" "K" "M" "O" "Q" "S" "U" "W"...
vowels chr [1:5] "A" "E" "I" "O" "U"
x num [1:5] 3.507 0.107 -0.693 -0.293 -1.093
Functions
Files Plots Packages Help Viewer Presentation
New Folder
New Blank File
Delete
Rename
More
Home
Name
Size
Modified
.RData 114 B Mar 20, 2023, 5:57 PM
.Rhistory 106 B Mar 20, 2023, 8:40 PM
1.Write a C program for Caesar cipher involves replacing each letter of t... 522 B Feb 1, 2023, 1:34 PM
1.Write a C program for Caesar cipher involves replacing each letter of t... 1.8 MB Feb 1, 2023, 1:34 PM
5. Construct a scheduling program with C that selects the waiting proce... 1.4 KB Jan 10, 2023, 2:58 PM
5. Construct a scheduling program with C that selects the waiting proce... 129.9 KB Jan 10, 2023, 2:04 PM
11.Illustrate the concept of multithreading using a C program.exe 184.5 KB Jan 10, 2023, 1:38 PM
12. Design a C program to simulate the concept of Dining-Philosophers... 187.7 KB Jan 11, 2023, 10:21 AM
13. Construct a C program for implementation the various memory allo... 129.2 KB Jan 11, 2023, 10:24 AM
14. Construct a C program to organize the file using single level directo... 130.2 KB Jan 11, 2023, 10:27 AM
14.Construct a C program to organize the file using single level director... 1.2 KB Jan 10, 2023, 1:49 PM
14.Construct a C program to organize the file using single level director... 130 KB Jan 10, 2023, 1:52 PM
15. Design a C program to organize the file using two level directory str... 130.7 KB Jan 11, 2023, 10:28 AM
16. Develop a C program for implementing random access file for proc... 128.4 KB Jan 11, 2023, 10:29 AM
17. Illustrate the deadlock avoidance concept by simulating Banker's al... 130.2 KB Jan 11, 2023, 10:32 AM
18 Construct a C program to simulate producer-consumer problem usi... 129.4 KB Jan 11, 2023, 10:33 AM
19. Design a C program to implement process synchronization using m... 183.6 KB Jan 11, 2023, 10:34 AM
21. Develop a C program to implement worst fit algorithm of memory ... 1.8 MB Jan 11, 2023, 10:38 AM
88°F
Partly sunny
Search
ENG
IN
11:30
21-03-2023