S.koushik

192121150

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.

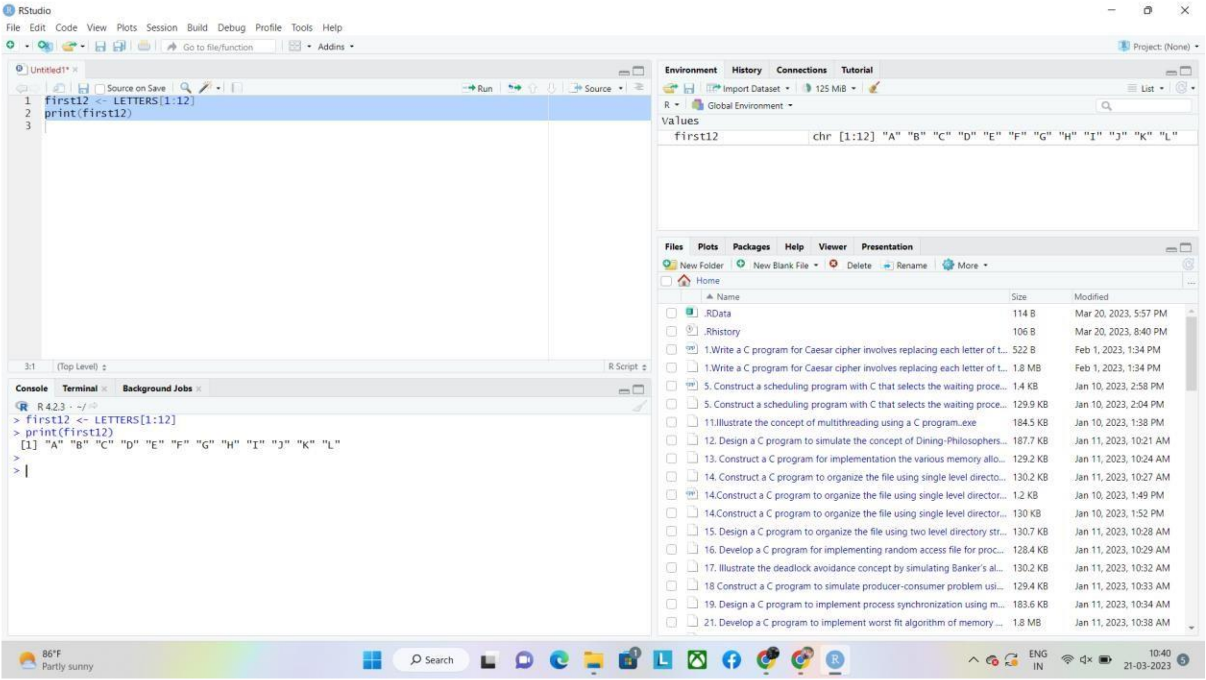
Program:

(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"
```

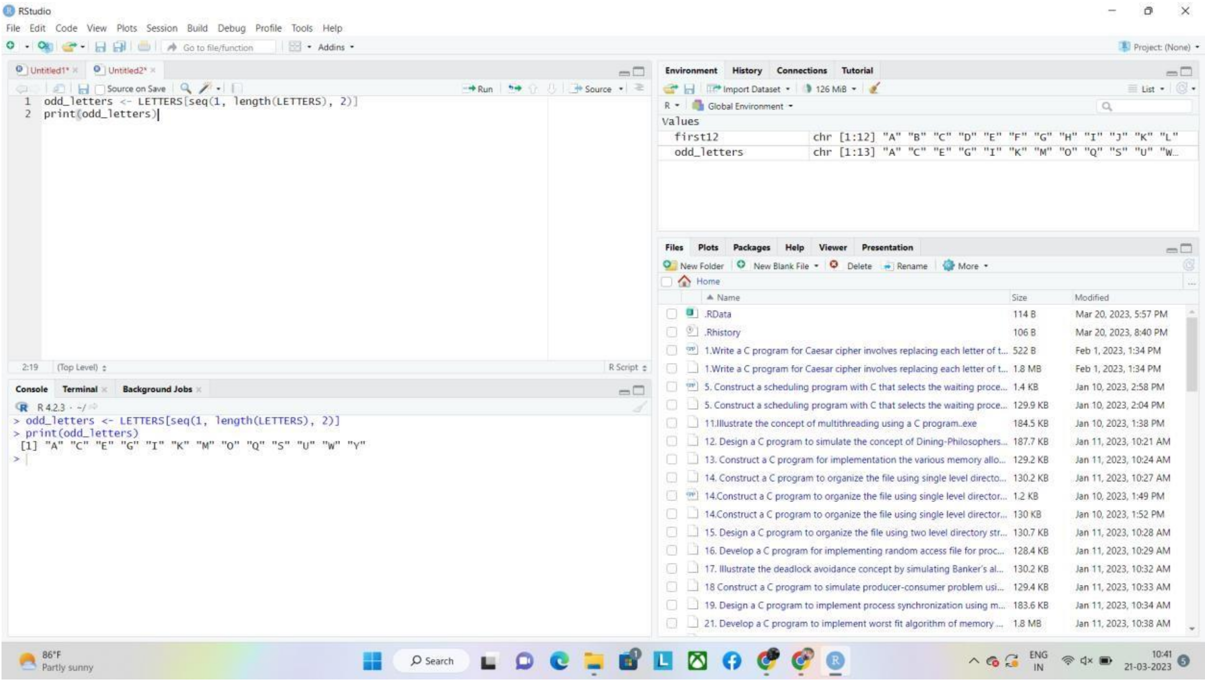


(ii) the odd 'numbered' letters; Program:

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:

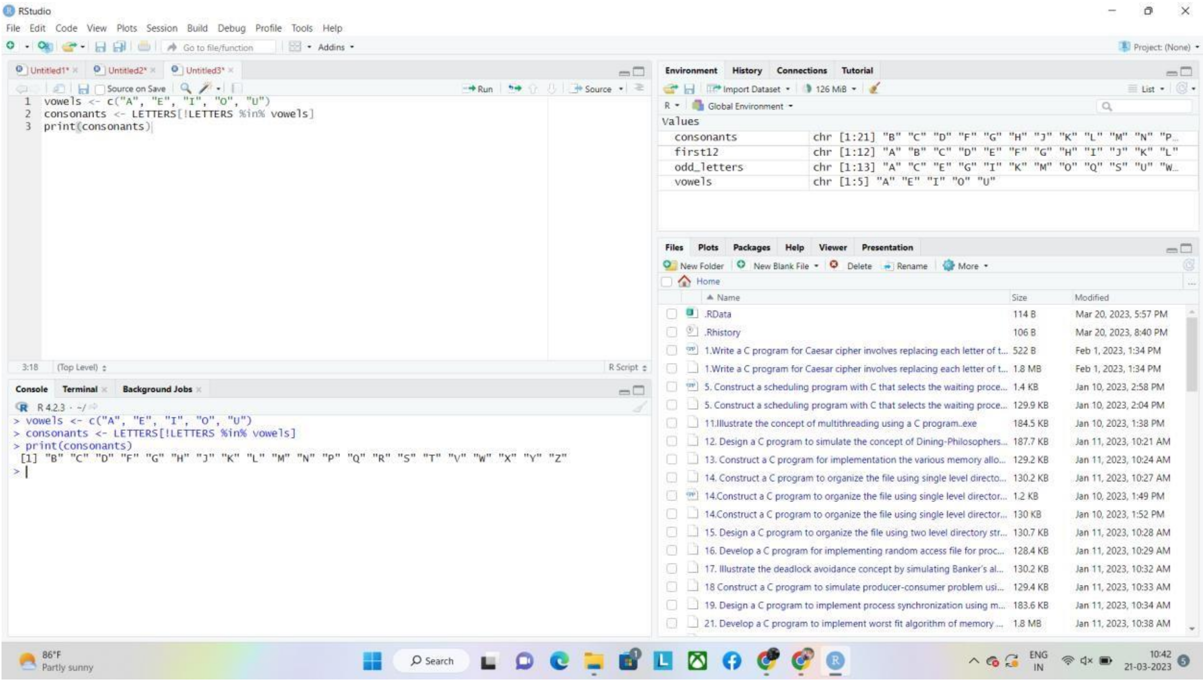
vowels <- c("A", "E", "I", "O", "U")

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"
"Z

>
```



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)

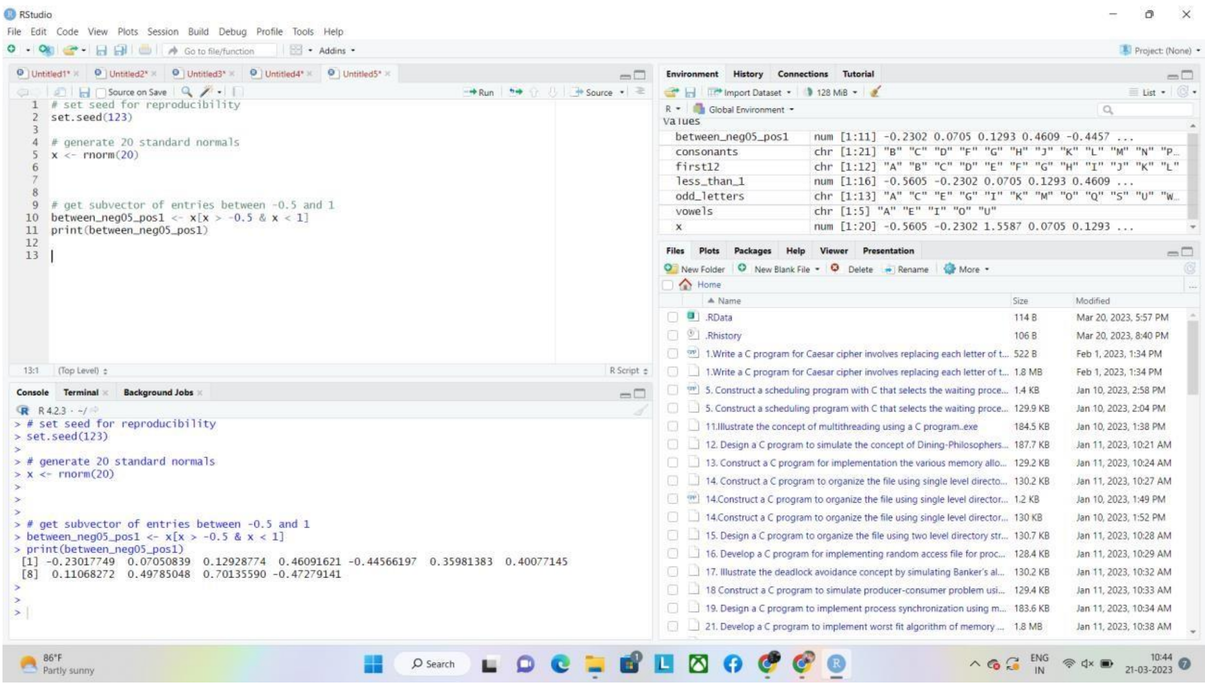
 # get subvector of entries in x which are less than 1

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 -
0.6868528
 [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

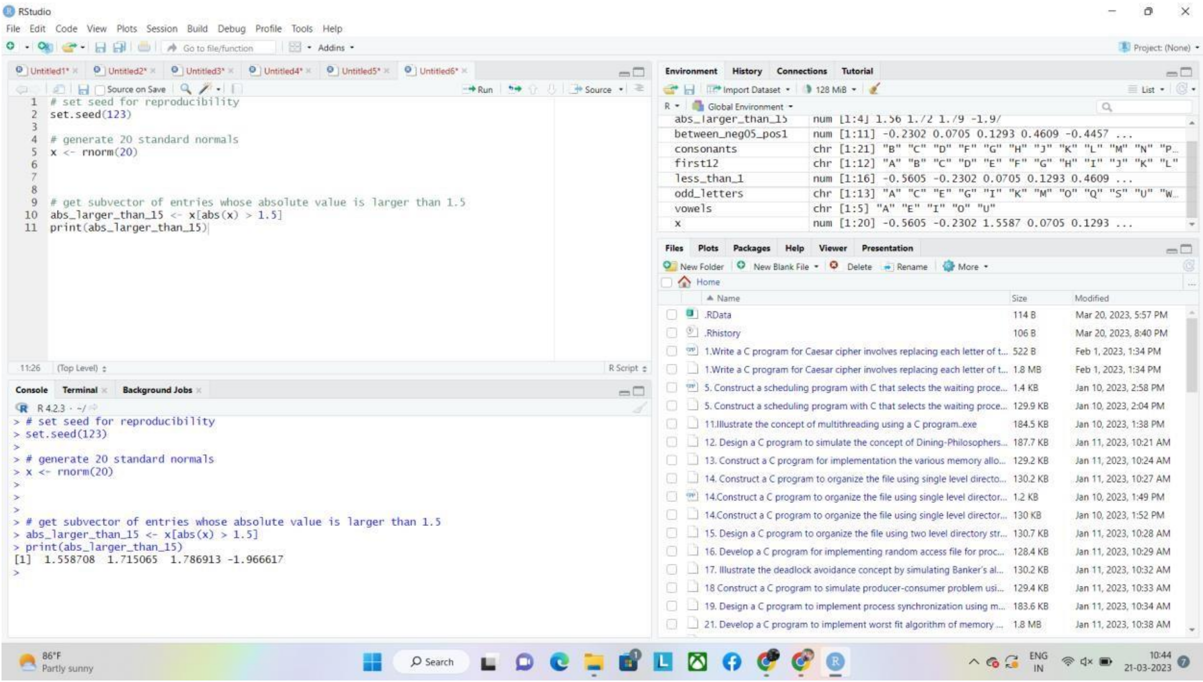
<- rnorm(20)

 # 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)

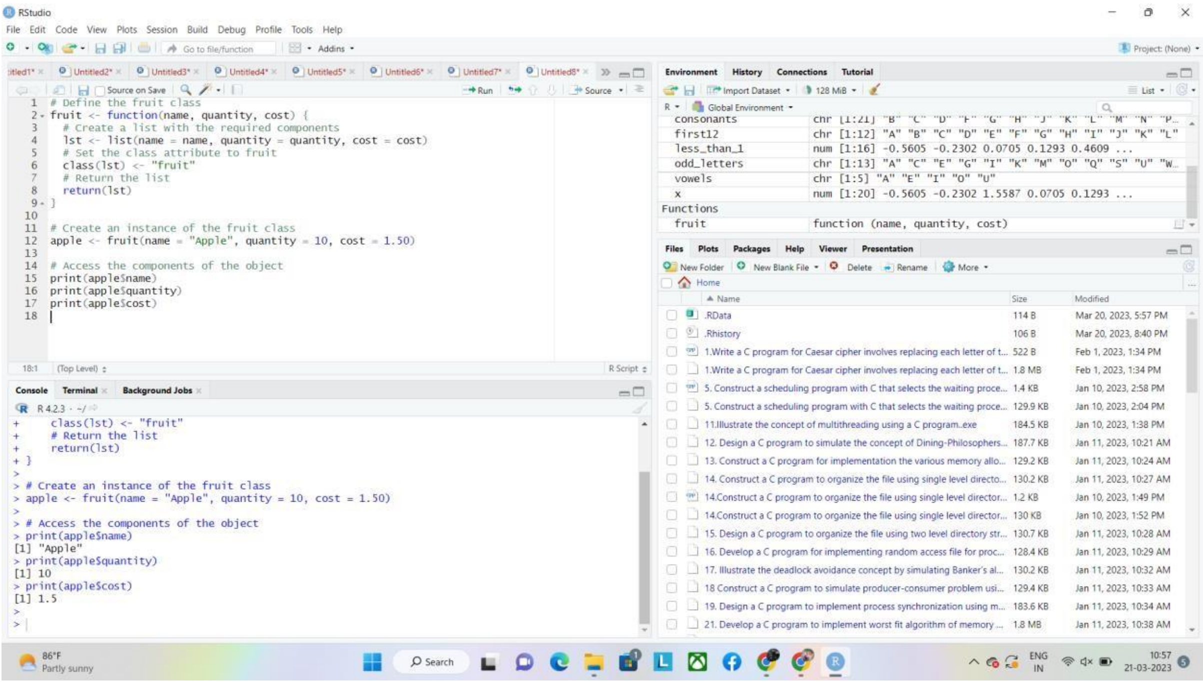
get subvector of entries whose absolute value is larger than 1.5

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

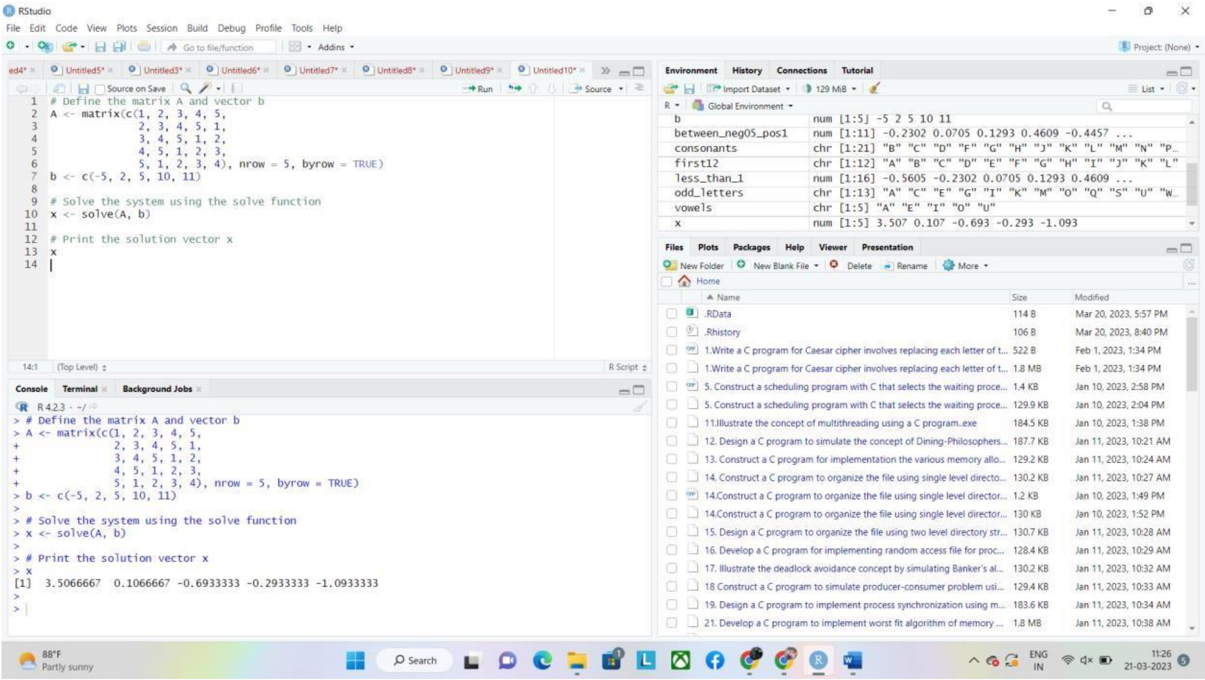
<- solve(A, b)

 # 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)

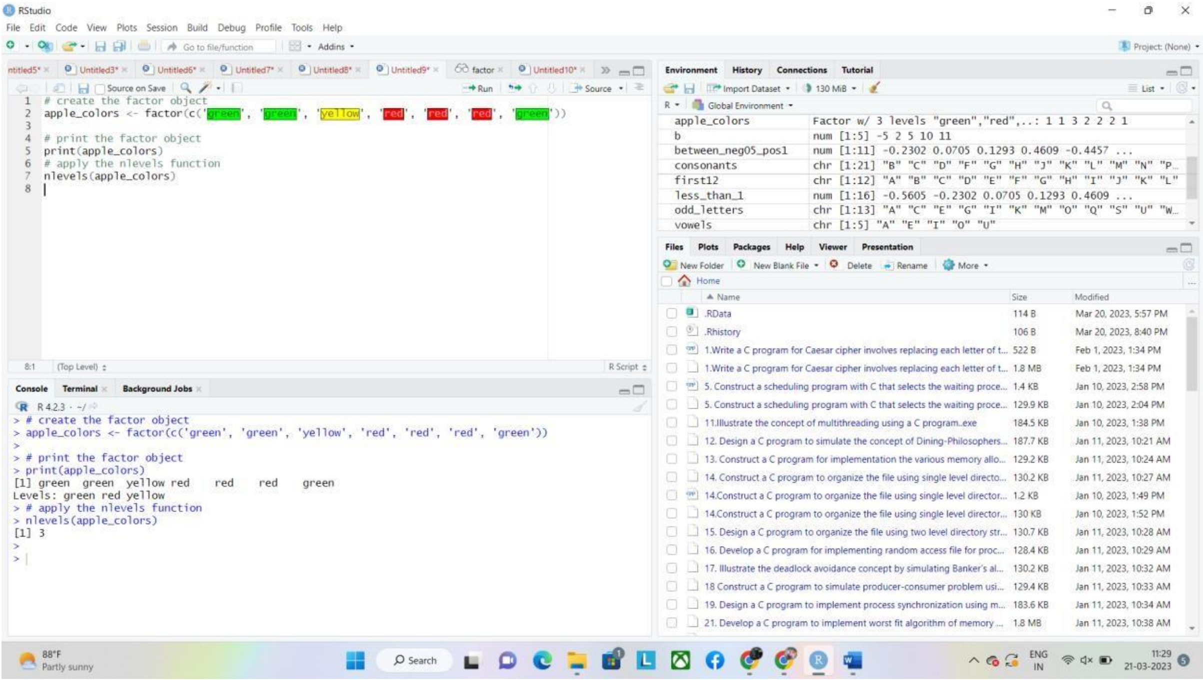
 # apply the nlevels function

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
 >

>
```