R – Programs

1.Factorial of a number

AIM:

To write a R program for factorial.

PROGRAM:

```
num = as.integer(readline(prompt="Enter a number: "))
factorial = 1
if(num < 0) {
print("Sorry, factorial does not exist for negative numbers")
} else if(num == 0) {
print("The factorial of 0 is 1")
} else {
for(i in 1:num) {
factorial = factorial * i
}
print(paste("The factorial of", num ,"is",factorial))
}
```

Output

```
Enter a number: 8

[1] "The factorial of 8 is 40320"
```

2.Take input from user

AIM:

To write a R program for input from user.

PROGRAM:

```
my.name <- readline(prompt="Enter name: ")

my.age <- readline(prompt="Enter age: ")

my.age <- as.integer(my.age)

print(paste("Hi,", my.name, "next year you will be", my.age+1, "years old."))
```

Output:

```
Enter name: Melvin

Enter age: 17

  [1] "Hi, Melvin next year you will be 18 years old."
```

3.To print Hello World.

AIM:

To write a R program to print hello world.

PROGRAM:

```
print("Hello World!")

Output [1] "Hello World!" # Quotes can be suppressed in the output

 print("Hello World!", quote = FALSE)

Output [1] Hello World!
```

```
print(paste("How","are","you?"))
```

Output:

```
[1] "How are you?"
```

4.Vector Elements Arithmetic

AIM:

To write a R program for Vector Elements Arithmetic.

PROGRAM:

```
sum(2,7,5)
```

Output - [1] 14

```
 x
```

Output [1] 2 NA 3 1 4

```
 sum(x)    # if any element is NA or NaN, result is NA or NaN
```

Output [1] NA

```
sum(x, na.rm=TRUE)    # this way we can ignore NA and NaN values
```

Output [1] 10

```
 mean(x, na.rm=TRUE)
```

Output [1] 2.5

```
prod(x, na.rm=TRUE)
```

```
Output [1] 24
```

5.R program to illustrate the use of Arithematic operators

AIM:

To write a R program to illustrate the uise of Arithematic operators.

PROGRAM:

```
vec1 <- c(0, 2)
vec2 <- c(2, 3)
cat ("Addition of vectors :", vec1 + vec2, "\n")
cat ("Subtraction of vectors :", vec1 - vec2, "\n")
cat ("Multiplication of vectors :", vec1 * vec2, "\n")
cat ("Division of vectors :", vec1 / vec2, "\n")
cat ("Modulo of vectors :", vec1 %% vec2, "\n")
cat ("Power operator :", vec1 ^ vec2)
```

Output:

```
Addition of vectors : 2 5
Subtraction of vectors : -2 -1
Multiplication of vectors : 0 6
Division of vectors : 0 0.6666667
Modulo of vectors : 0 2
Power operator : 0 8
```

6.R program to illustrate the use of Logical operators

AIM:

To write a R program to illustrate the use of logical operators.

PROGRAM:

```
vec1 <- c(0,2)
vec2 <- c(TRUE,FALSE)
cat ("Element wise AND :", vec1 & vec2, "\n")
cat ("Element wise OR :", vec1 | vec2, "\n")
cat ("Logical AND :", vec1 && vec2, "\n")
cat ("Logical OR :", vec1 || vec2, "\n")
cat ("Negation :", !vec1)
```

Output:

```
Element wise AND : FALSE FALSE
Element wise OR : TRUE TRUE
Logical AND : FALSE
Logical OR : TRUE
Negation : TRUE FALSE
```

7.R program to illustrate the use of Relational operators

AIM:

To write a R program to illustrate the use of relational operators.

PROGRAM:

```
vec1 <- c(0, 2)
vec2 <- c(2, 3)
cat ("Vector1 less than Vector2 :", vec1 < vec2, "\n")
```

```
cat ("Vector1 less than equal to Vector2 :", vec1 <= vec2, "\n")
cat ("Vector1 greater than Vector2 :", vec1 > vec2, "\n")
cat ("Vector1 greater than equal to Vector2 :", vec1 >= vec2, "\n")
cat ("Vector1 not equal to Vector2 :", vec1 != vec2, "\n")
```

Output:

```
Vector1 less than Vector2 : TRUE TRUE
Vector1 less than equal to Vector2 : TRUE TRUE
Vector1 greater than Vector2 : FALSE FALSE
Vector1 greater than equal to Vector2 : FALSE FALSE
Vector1 not equal to Vector2 : TRUE TRUE
```

8.R program to illustrate the use of Assignment operators

AIM:

To write a R program to illustrate the use of assignment operators.

PROGRAM:

```
vec1 <- c(2:5)
c(2:5) ->> vec2
vec3 <<- c(2:5)
vec4 = c(2:5)
c(2:5) -> vec5

cat ("vector 1 :", vec1, "\n")
cat("vector 2 :", vec2, "\n")
cat ("vector 3 :", vec3, "\n")
```

```
cat("vector 4 :", vec4, "\n")

cat("vector 5 :", vec5)
```

Output:

```
vector 1 : 2 3 4 5
vector 2 : 2 3 4 5
vector 3 : 2 3 4 5
vector 4 : 2 3 4 5
vector 5 : 2 3 4 5
```

9.R program to illustrate the use of Miscellaneous operators

AIM:

To write a R program to illustrate the use of miscellaneous.

PROGRAM:

```
mat <- matrix (1:4, nrow = 1, ncol = 4)

print("Matrix elements using : ")

print(mat)

product = mat %*% t(mat)

print("Product of matrices")

print(product,)

cat ("does 1 exist in prod matrix :", "1" %in% product)
```

Output:

```
[1] "Matrix elements using : "
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4

[1] "Product of matrices"
     [,1]
[1,]   30

does 1 exist in prod matrix : FALSE
```

10.R Program to check if the input number is odd or even.

AIM:

To write a R program to check if the input number is odd or even.

PROGRAM:

```
num = as.integer(readline(prompt="Enter a number: "))

if((num %% 2) == 0) {

print(paste(num,"is Even"))

} else {

print(paste(num,"is Odd"))

}
```

Output 1

```
Enter a number: 189
```

```
[1] "189 is Odd"
```

Output 2

```
Enter a number: 10


[1] "10 is Even"
```

11.Fibonacci Series

AIM:

To write a R program for fibonacci series.

PROGRAM:

```
nterms = as.integer(readline(prompt="How many terms? "))
n1 = 0
n2 = 1
count = 2
if(nterms <= 0)
      {
       print("Plese enter a positive integer")
         }
else {
       if(nterms == 1)
 {
print("Fibonacci sequence:")
print(n1)
} else {
print("Fibonacci sequence:")
print(n1)
print(n2)
while(count < nterms) {
nth = n1 + n2
print(nth)
n1 = n2
n2 = nth
count = count + 1
}
```

```
}
}
```

Output

```
How many terms? 7
[1] "Fibonacci sequence:"
[1] 0
[1] 1
[1] 1
[1] 2
[1] 3
[1] 5
[1] 8
```

12.R Program to display the Fibonacci sequence up to n-th term using recursive functions

AIM:

To write a R program to display the fibonacci sequence up to n-th term using recursive function.

PROGRAM:

```
recurse_fibonacci <- function(n) {
if(n <= 1) {
return(n)
} else {
return(recurse_fibonacci(n-1) + recurse_fibonacci(n-2))
}
}
nterms = as.integer(readline(prompt="How many terms? "))
if(nterms <= 0) {
print("Plese enter a positive integer")
} else {
print("Fibonacci sequence:")
for(i in 0:(nterms-1)) {
print(recurse_fibonacci(i))
}
}
```

Output

```
How many terms? 9
[1] "Fibonacci sequence:"
[1] 0
[1] 1
[1] 1
[1] 2
[1] 3
[1] 5
[1] 8
[1] 13
[1] 21
```

13.To Print the factor of a number

AIM:

To write a R program to print the factor of a number.

PROGRAM:

```
print_factors <- function(x) {

print(paste("The factors of",x,"are:"))

for(i in 1:x) {

if((x %% i) == 0) {

print(i)

}

}

}
```

Output

```
> print_factors(120)

[1] "The factors of 120 are:"
```

[1] 1

[1] 2

[1] 3

[1] 4

[1] 5

[1] 6

[1] 8

[1] 10

[1] 12

[1] 15

[1] 20

[1] 24

[1] 30

[1] 40

[1] 60

[1] 120

14.R Program make a simple calculator that can add, subtract, multiply and divide using functions

AIM:

To write a R program make a simple calculator that can add, subtract, multiply and divide using functions.

PROGRAM:

```
add <- function(x, y) {

return(x + y)
```

```
}
subtract <- function(x, y) {
return(x - y)
}
multiply <- function(x, y) {
return(x * y)
}
divide <- function(x, y) {
return(x / y)
}
print("Select operation.")
print("1.Add")
print("2.Subtract")
print("3.Multiply")
print("4.Divide")
choice = as.integer(readline(prompt="Enter choice[1/2/3/4]: "))
num1 = as.integer(readline(prompt="Enter first number: "))
num2 = as.integer(readline(prompt="Enter second number: "))
operator <- switch(choice,"+","-","*","/")
result <- switch(choice, add(num1, num2), subtract(num1, num2), multiply(num1, num2), divide(num1, num2))
print(paste(num1, operator, num2, "=", result))
```

Output

```
[1] "Select operation."
[1] "1.Add"
[1] "2.Subtract"
[1] "3.Multiply"
[1] "4.Divide"
```

Enter choice[1/2/3/4]: 4

Enter first number: 20

Enter second number: 4

[1] "20 / 4 = 5"

15.R Program to find the L.C.M. of two input number

AIM:

To write a R program to find the L.C.M. of two input number.

PROGRAM:

```
1cm <- function(x, y) {
if(x > y) {
greater = x
} else {
greater = y
}
while(TRUE) {
if((greater %% x == 0) && (greater %% y == 0)) {
lcm = greater
break
}
greater = greater + 1
}
return(lcm)
}
```

```
num1 = as.integer(readline(prompt = "Enter first number: "))
num2 = as.integer(readline(prompt = "Enter second number: "))
print(paste("The L.C.M. of", num1,"and", num2,"is", lcm(num1, num2)))
```

Output

```
Enter first number: 24
Enter second number: 25
[1] "The L.C.M. of 24 and 25 is 600"
```

16.R program to create numeric Vectors

AIM:

To write a R program to create numeric vectors.

PROGRAM:

```
v1 <- c(4, 5, 6, 7)
typeof(v1)
v2 <- c(1L, 4L, 2L, 5L)
typeof(v2)
```

Output:

```
[1] "double"
[1] "integer"
```

17.R program to access elements of a Vector

AIM:

To write a R program to access element 0f a vector.

PROGRAM:

```
X <- c(2, 5, 18, 1, 12)
cat('Using Subscript operator', X[2], '\n')
Y <- c(4, 8, 2, 1, 17)
cat('Using combine() function', Y[c(4, 1)], '\n')
Z <- c(5, 2, 1, 4, 4, 3)
cat('Using Logical indexing', Z[Z>4])
```

Output

```
Using Subscript operator 5
Using combine() function 1 4
Using Logical indexing 5
```

18.R program to modify elements of a Vector

AIM:

To write a R program to modify elements of a vector.

PROGRAM:

```
X <- c(2, 7, 9, 7, 8, 2)
X[3] <- 1
X[2] <-9
cat('subscript operator', X, '\n')
X[X>5] <- 0
cat('Logical indexing', X, '\n')
X <- X[c(3, 2, 1)]
cat('combine() function', X)
```

Output

```
subscript operator 2 9 1 7 8 2
Logical indexing   2 0 1 0 0 2
combine() function 1 0 2
```

19.R program to delete a Vector

AIM:

To write a R program to delete a vector.

PROGRAM:

```
M <- c(8, 10, 2, 5)
M <- NULL
cat('Output vector', M)
```

Output:

```
Output vector NULL
```

20.R program to sort elements of a Vector

AIM:

To write a R program to sort element of a vector.

PROGRAM:

```
X <- c(8, 2, 7, 1, 11, 2)
A <- sort(X)
cat('ascending order', A, '\n')
```

```
B <- sort(X, decreasing = TRUE)
cat('descending order', B)
```

Output:

```
ascending order 1  2  2  7  8 11
descending order 11  8  7  2  2  1
```