R PROGRAMMING

DAY 2 LAB MANUAL

P.Barath

192121147

IMPLEMENTATION OF VECTOR RECYCLING, APPLY FAMILY &amp; RECURSION

1.Demonstrate Vector Recycling in R.

PROGRAM:

```
vec1=1:6

vec2=1:2

print(vec1+vec2)
```

OUTPUT:

```
> vec1=1:6

> vec2=1:2

> print(vec1+vec2)

[1] 2 4 4 6 6 8
```

2.Demonstrate the usage of apply function in R

program:

```
m1 <- matrix(C<-(1:10),nrow=5, ncol=6)

m1
```

```
a_m1 <- apply(m1, 2, sum)
a_m1
```

output;

```
> m1 <- matrix(C<-(1:10),nrow=5, ncol=6)
> m1
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    6    1    6    1    6
[2,]    2    7    2    7    2    7
[3,]    3    8    3    8    3    8
[4,]    4    9    4    9    4    9
[5,]    5   10    5   10    5   10
> a_m1 <- apply(m1, 2, sum)
> a_m1
```

3.Demonstrate the usage of lapply function in R

program:

```
names <- c("dhanush","barath","kumar","kaja","sudhan")
print("original data:")
names

print("data after lapply():")
lapply(names,toupper)
```

output:

```
> names <- c("dhanush","barath","kumar","kaja","sudhan")
> print("original data:")

[1] "original data:"

> names

[1] "dhanush" "barath"  "kumar"   "kaja"

[5] "sudhan"

>

> print("data after lapply():")

[1] "data after lapply():"

> lapply(names,toupper)

[[1]]

[1] "DHANUSH"


[[2]]

[1] "BARATH"


[[3]]

[1] "KUMAR"


[[4]]

[1] "KAJA"


[[5]]

[1] "SUDHAN"


>
```

4.Demonstrate the usage of sapply function in R

program:

```
dt <- cars
lmn_cars <- lapply(dt, min)
smn_cars <- sapply(dt, min)
lmn_cars
```

output:

```
> dt <- cars
> lmn_cars <- lapply(dt, min)
> smn_cars <- sapply(dt, min)
> lmn_cars
$speed
[1] 4

$dist
[1] 2
```

5.Demonstrate the usage of tapply function in r

program:

```
data(iris)
tapply(iris$Sepal.Width, iris$Species, median)
```

output:

```
> data(iris)
> tapply(iris$Sepal.Width, iris$Species, median)
    setosa versicolor  virginica
       3.4        2.8        3.0
>
```

6.Demonstrate the usage of mapply function in R

program:

```
list(rep(1, 5), rep(2, 4), rep(3, 3), rep(4, 2), rep(5,1))
```

output:

```
> list(rep(1, 5), rep(2, 4), rep(3, 3), rep(4, 2), rep(5,1))
[[1]]
[1] 1 1 1 1 1

[[2]]
[1] 2 2 2 2

[[3]]
[1] 3 3 3

[[4]]
[1] 4 4

[[5]]
[1] 5

>
```

7.Sum of Natural Numbers using Recursion

program;

```
sum_natural_numbers <- function(n) {
  if (n == 1) {
    return(1)
  } else {
    return(n + sum_natural_numbers(n-1))
  }
}

# Example usage:
sum_natural_numbers(5) # Returns 15
```

output:

```
 sum_natural_numbers <- function(n) {
+     if (n == 1) {
+         return(1)
+     } else {
+         return(n + sum_natural_numbers(n-1))
+     }
+ }
>
> # Example usage:
> sum_natural_numbers(5) # Returns 15
[1] 15
>
```

8.Write a program to generate Fibonacci sequence using Recursion in R

program:

```
fibonacci <- function(n) {
  if (n <= 1) {
    return(n)
  } else {
    return(fibonacci(n-1) + fibonacci(n-2))
  }
}

# Example usage:
for (i in 0:10) {
  cat(fibonacci(i), " ")
}
```

output:

```
> fibonacci <- function(n) {
+     if (n <= 1) {
+         return(n)
+     } else {
+         return(fibonacci(n-1) + fibonacci(n-2))
+     }
+ }
> 
> # Example usage:
> for (i in 0:10) {
+     cat(fibonacci(i), " ")
+ }
0  1  1  2  3  5  8  13  21  34  55  >
```

9.Write a program to find factorial of a number in R using recursion.

program:

```
factorial <- function(n) {
  if (n == 0) {
    return(1)
  } else {
    return(n * factorial(n - 1))
  }
}

# Example usage:
factorial(5) # Returns 120
```

output;

```
> factorial <- function(n) {
+     if (n == 0) {
+         return(1)
+     } else {
+         return(n * factorial(n - 1))
+     }
+ }
>
> # Example usage:
> factorial(5) # Returns 120
[1] 120
>
>
```