FINDING SUM, MEAN, PRODUCT & ADDITION OF VECTORS

AIM:

To write R program to find Sum, Mean and Product of a Vector

PROGRAM:

```
x = c(10, 20, 30)
print("Sum:")
print(sum(x))
print("Mean:")
print(mean(x))
print("Product:")
print(prod(x))
y=c(1,2,3)
print(paste(x,”\n”,y,”\nAddition of two vectors”)
print(x+y)
```

RESULT:

The above program is executed successfully.

OUTPUT:

```
[1] "Sum:"
[1] 60
[1] "Mean:"
[1] 20
[1] "Product:"
[1] 6000
10 20 30
1  2  3
Addition of two vectors
[1] 11 22 33
```

GENERATE RANDOM NUMBER FROM STANDARD DISTRIBUTIONS

AIM:

To write R program to generate random numbers from standard distribution

PROGRAM:

```
randomsd->function(length,start,end){
rnorm(length,start,end)
}
length=as.integer(readline(prompt="Enter the length :- "))
start=as.integer(readline(prompt="Enter the starting point :- "))
end=as.integer(readline(prompt="Enter the ending point :- "))
randomsd(length,start,end)
```

RESULT:

The above program is executed successfully.

OUTPUT:

```
    Enter the length :- 10
    Enter the starting point :- 10
    Enter the ending point :- 20
    [1]  13.4883302 -22.0229719  28.0231140
11.5093660  22.6994037  32.1992925
    [7] -22.0536870  -0.6583609  15.6940226
28.3465342
```

SAMPLE FROM A POPULATION

AIM:

To write r program to take sample from the population.

PROGRAM:

```
samplepop <- function(data,size){
print(sample(data,size))
}
data=c(23,45,21,34,5,6,7,8,86,45,3)
size=as.integer(readline(prompt="Enter the size :- "))
samplepop(data,size)
```

RESULT:

The above program is executed successfully.

OUTPUT:

```
Enter the size :- 4
[1] 45  5  5  3
```

FIND MIN MAX & SORT A VECTOR

AIM:

To write an R program to find minimum , maximum element & sort a vector.

PROGRAM:

```
# Creating a vector
x <- c(7, 4, 3, 9, 1.2, -4, -5, -8, 6)
print(x)
print("Minimum is")
print(min(x))
print("Maximum is")
print(max(x))
print("Sorted")
sort(x)
```

RESULT:

The above program is executed successfully.

OUTPUT:

```
[1]  7.0  4.0  3.0  9.0  1.2 -4.0 -5.0 -8.0  6.0
Minimum is
-8
Maximum is
9
Sorted
[1] -8.0 -5.0 -4.0  1.2  3.0  4.0  6.0  7.0  9.0
```

FACTORIAL

AIM:

To write an R program to find the factorial of a number.

PROGRAM:

```
facto <- function(){
n=as.integer(readline(prompt = ‘Input a num to factorial :’))
fact=1
if (n<0){
      cat(“Cannot give factorial”)
} else if(n=0){
      cat(“Factorial is 1”)
} else {
      for(i in 1:n){
      fact=fact*i}
      cat(“Factorial of”,n,”is”,fact)
}
}
facto()
```

RESULT:

The above program is executed successfully.

OUTPUT:

```
Input a num to factorial:5
[1] “Factorial of 5 is 120”
```

MULTIPLICATION TABLE

AIM:

To write an R program to print multiplication table of a number.

PROGRAM:

```
mult.tab <- function(x,y){
for( t in 1:y)
{
   cat( x, '*', t, '=', x* t,”\n”))
}
}
num= as.integer(readline(prompt=”Enter No for Multiplication table:”)
tab= as.integer(readline(prompt=”Enter No of times to be multiplied:”)
mult.tab(num,tab)
```

RESULT:

The above program is executed successfully.

OUTPUT:

```
Enter No for Multiplication table:3
Enter No of times to be multiplied:10
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
3 * 10 = 30
```

PRIME NUMBER

AIM:
To write an R program to check if a number is prime or not.

PROGRAM:

```
num = as.integer(readline(prompt="Enter a number: "))
flag = 0
if(num > 1) {
      flag = 1
      for(i in 2:(num-1)) {
            if ((num %% i) == 0) {
                  flag = 0
                  break }
            }
}
if(num == 2)    flag = 1
if(flag == 1) {
print(paste(num,"is a prime number"))
} else {
print(paste(num,"is not a prime number"))
}
```

RESULT:
The above program is executed successfully.

OUTPUT:

```
Enter a number:69
[1] "69 is not a prime number"
```

ARMSTRONG NUMBER

AIM:

To write an R program to check if a number is armstrong number or not.

PROGRAM:

```
num = as.integer(readline(prompt="Enter a number: "))
sum = 0
temp = num
while(temp > 0) {
digit = temp %% 10
sum = sum + (digit ^ 3)
temp = floor(temp / 10)
}
if(num == sum) {
print(paste(num, "is an Armstrong number"))
} else {
print(paste(num, "is not an Armstrong number"))
}
```

RESULT:

The above program is executed successfully.

Output:

```
Enter a number: 23
[1]"23 is not an Armstrong number"
```

FIBONACCI SEQUENCE

AIM:

To write an R program to print Fibonacci sequence up to a given number.

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
}}}
```

RESULT:

The above program is executed successfully.

OUTPUT:

```
How many terms?10
[1] "Fibonacci sequence:"
[1] 0
[1] 1
[1] 1
[1] 2
[1] 3
[1] 5
[1] 8
[1] 13
[1] 21
[1] 34
```

LEAP YEAR

AIM:

To write an R program to find if a year is a leap year.

PROGRAM:

```
year = as.integer(readline(prompt="Enter a year: "))
if((year %% 4) == 0) {
if((year %% 100) == 0) {
if((year %% 400) == 0) {
print(paste(year,"is a leap year"))
} else {
print(paste(year,"is not a leap year"))
}
} else {
print(paste(year,"is a leap year"))
}
} else {
print(paste(year,"is not a leap year"))
}
```

RESULT:

The above program is executed successfully.

OUTPUT:

```
Enter a year: 1900
[1] "1900 is not a leap year"
```

DATAFRAME

AIM:

To write an R program to create and print dataframe.

PROGRAM:

```
x <- data.frame("SN" = 1:2, "Age" = c(21,15), "Name" =
c("John","Dora"),stringsAsFactors=FALSE)
str(x)
x["Name"]
print(“Adding a row of data”)
rbind(x,list(1,16,"Paul"))
```

RESULT:

The above program is executed successfully.

OUTPUT:

```
'data.frame':  2obs. of  3variables:
$ SN  :int  1 2
 $ Age :num  21 15
 $ Name:chr  "John" "Dora"
Name
1 John
2 Dora[1] "Adding a row of data"
  SN Age Name
1  1  21 John
2  2  15 Dora
3  1  16 Paul
```