1. R PROGRAM TO MAKE A SIMPLE CALCULATOR

```
add=function(x,y){
  return(x+y)
}
sub=function(x,y){
  return(x-y)
}
mul=function(x,y){
  return(x*y)
}
div=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),sub(num1,num2),mul(num1,num2),div(
num1,num2))
print(paste(num1,operator,num2,"=",result))
```

OUT-PUT

[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”

2. R PROGRAM TO FIND THE SUM OF NATURAL

```
 num = as.integer(readline(prompt = "Enter a number: "))
 if(num < 0)
 {
 print("Enter a positive number")
 }
else {
    sum = 0
    while(num > 0) {
    sum = sum + num
    num = num - 1
 }
      print(paste("The sum is", sum))
}
```

OUT-PUT

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

Enter number: 10

[1] is sum “55”

3. R PROGRAM TO FIND HCF OR GCD

```
hcf <- function(x, y) {
# choose the smaller number
if(x > y) {
smaller = y
} else {
smaller = x
}
for(i in 1:smaller) {
if((x %% i == 0) && (y %% i == 0)) {
hcf = i
}
}
return(hcf)
}
# take input from the user
num1 = as.integer(readline(prompt = "Enter first number: "))
num2 = as.integer(readline(prompt = "Enter second number: "))
print(paste("The H.C.F. of", num1,"and", num2,"is", hcf(num1, num2)))
```

OUT-PUT

```
# take input from the user
Enter first number: 72
Enter second number: 120
 [1] "The H.C.F. of 72 and 120 is 24"
```

4. R PROGRAM TO FIND THE FACTORS OF A NUMBER

```
print_factors <- function(x) {
print(paste("The factors of",x,"are:"))
for(i in 1:x) {
if((x %% i) == 0) {
print(i)
}
}
}
```

OUT-PUT

```
> 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
```

5. R PROGRAM TO PRINT THE FIBONACCI SEQUENCE

```
nterms = as.integer(readline(prompt="How many terms? "))
# first two terms
n1 = 0
n2 = 1
count = 2
# check if the number of terms is valid
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)
# update values
n1 = n2
n2 = nth
count = count + 1
}
}
}
```

OUT-PUT

```
nterms = as.integer(readline(prompt="How many terms? "))
How many terms? 7

 [1] “Fibonacci sequence :”
[1] 0
[1] 1
[1] 2
[1] 3
[1] 5
[1] 8
```

6. R PROGRAM TO TAKE INPUT FROM USER

```
my.name <- readline(prompt="Enter name: ")
my.age <- readline(prompt="Enter age: ")
# convert character into integer
my.age <- as.integer(my.age)
print(paste("Hi,", my.name, "next year you will be", my.age+1, "years old."))
```

OUT-PUT

```
Enter name: venkatesh
Enter age: 30
[1] "Hi, venkatesh next year you will be 31 years old."
```

7. R PROGRAM TO FIND MINIMUM AND MAXIMUM.

```
> x<-c(5,8,3,9,2,7,4,6,10)
>x
[1] 5  8  3  9  2  7  4  6 10
> # find value at minimum index
> which.min(x)
[1] 5
> # find value at maximum index
> which.max(x)
[1] 9
> # alternate way to find the minimum
> x[which.min(x)]
[1] 2
> # find the minimum
> min(x)
[1] 2
> # find the maximum
> max(x)
[1] 10
> # find the range
> range(x)
[1]  2 10
```

8. R PROGRAM TO CHECK FOR LEAP YEAR.

```
year=as.integer(readline(prompt="enter a year:"))
if((year%%4)==0){
if((year%%100)==0){
if((year%%1000)==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,"isnot a leap year"))
}
```

OUT-PUT

year=as.integer(readline(prompt="enter a year:"))

2024

2024 is leap year

9. R PROGRAM TO CHECK FOR PRIME OR NOT

```
# Program to check if the input number is prime or not
# take input from the user
num = as.integer(readline(prompt="Enter a number: "))
flag = 0
# prime numbers are greater than 1
if(num > 1) {
# check for factors
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"))
}
```

OUT-PUT

```
num = as.integer(readline(prompt="Enter a number: "))
Enter number : 23
23 is a prime number
```

10. R PROGRAM TO CHECK A NUMBER IS ARMSTRONG OR NOT.

```
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"))
}
```

OUT-PUT

Enter a number 153

[1] “153” is an a Armstrong number

11. R PROGRAM TO SORT A VECTOR

```
> x<-c(7,1,8,3,2,6,5,2,2,4)

> x

[1] 7 1 8 3 2 6 5 2 2 4

> # sort in ascending order

> sort(x)

[1] 1 2 2 2 3 4 5 6 7 8

> # sort in descending order

> sort(x, decreasing=TRUE)

[1] 8 7 6 5 4 3 2 2 2 1

> # vector x remains unaffected

> x

[1] 7 1 8 3 2 6 5 2 2 4

> order(x)

[1]  2  5  8  9  4 10  7  6  1  3

> order(x, decreasing=TRUE)

[1]  3  1  6  7 10  4  5  8  9  2

> x[order(x)]    # this will also sort x

[1] 1 2 2 2 3 4 5 6 7 8
```

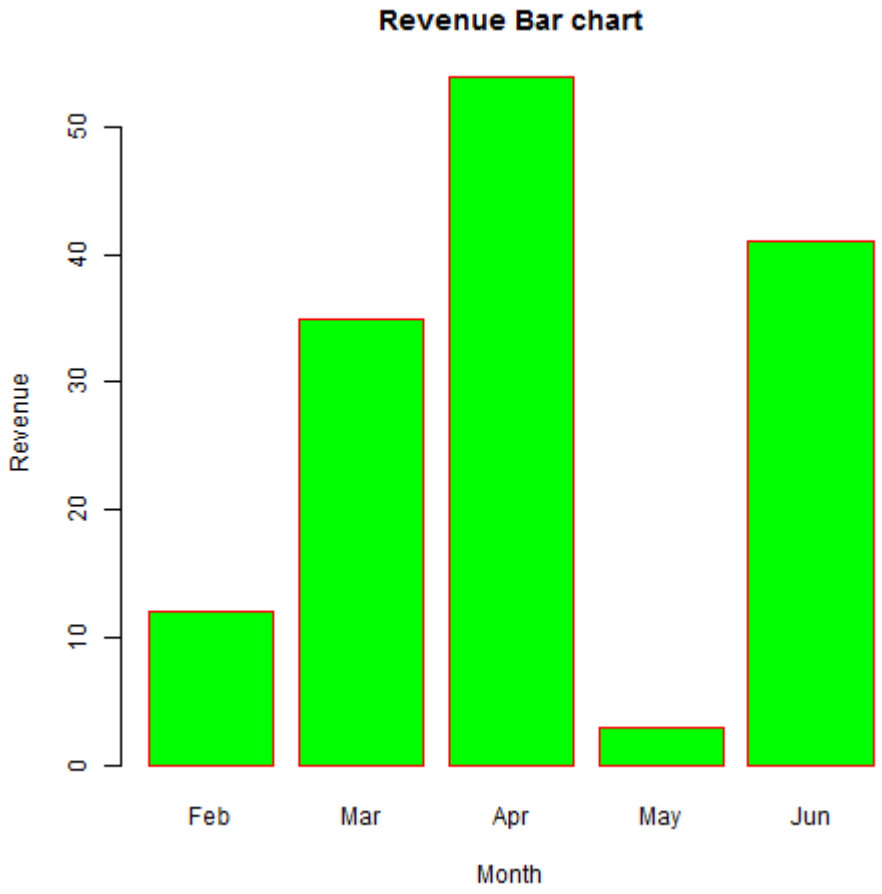
12. R PROGRAM TO CREATING THE DATA FOR BAR CHART

```
H <- c(12,35,54,3,41)
M<- c("Feb","Mar","Apr","May","Jun")

png(file = "bar_properties.png")

# Plotting the bar chart
barplot(H,names.arg=M,xlab="Month",ylab="Revenue",col="Green",
      main="Revenue Bar chart",border="red")
# Saving the file
dev.off()
```

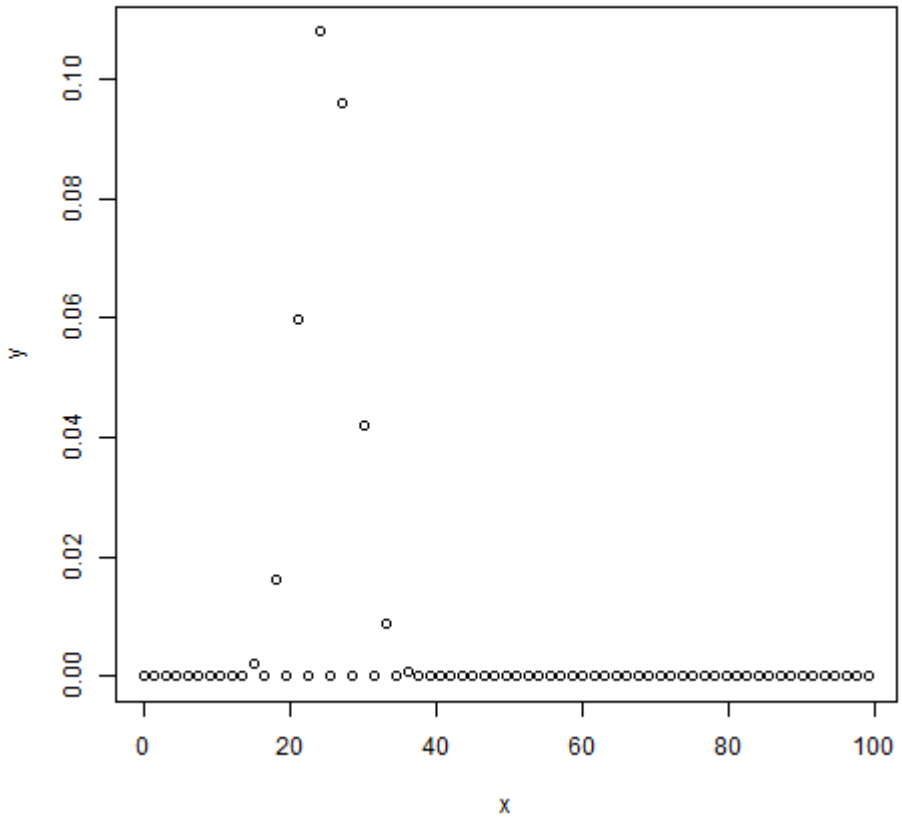
OUT-PUT


Revenue Bar chart
Revenue
0
10
20
30
40
50
Feb
Mar
Apr
May
Jun
Month

13.CREATING A SAMPLE OF 100 NUMBERS WHICH ARE INCREME NTED BY 1.5

```
x <- seq(0,100,by = 1.5)
# Creating the binomial distribution.
y <- dbinom(x,50,0.5)
 Giving a name to the chart file.
png(file = "dbinom.png")
# Plotting the graph.
plot(x,y)
# Saving the file.
dev.off()
```

OUT-PUT



14. CREATING INPUT VECTOR FOR LM() FUNCTION

```
x <- c(141, 134, 178, 156, 108, 116, 119, 143, 162, 130)
y <- c(62, 85, 56, 21, 47, 17, 76, 92, 62, 58)

# Applying the lm() function.
relationship_model<- lm(y~x)

#Printing the coefficient
print(summary(relationship_model))
```

OUT-PUT

```
Call:
lm(formula = y ~ x)

Residuals:
    Min      1Q  Median      3Q     Max
-38.948  -7.390   1.869  15.933  34.087

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 47.50833   55.18118   0.861    0.414
x            0.07276    0.39342   0.185    0.858

Residual standard error: 25.96 on 8 degrees of freedom
Multiple R-squared:  0.004257,  Adjusted R-squared:  -0.1202
F-statistic: 0.0342 on 1 and 8 DF,  p-value: 0.8579
```

15. R PROGRAM TO FIND THE FREQUENCY OF A DIGIT IN THE NUMBER.

```
num = as.integer(readline(prompt="Enter a number: "))
digit = as.integer(readline(prompt="Enter digit: "))
n=num
count = 0
while(num > 0) {
      if(num%%10==digit){
         countcount=count+1
      }
      num=as.integer(num/10)
}
print(paste("The frequency of",digit,"in",n,"is=",count))
```

OUT-PUT

Enter a number 1211436221

[1] “the frequency of 1 in 1211436221 is =4”