1.

```
prompt Enter value of x:
accept x
prompt Enter value of y:
accept y
create table Circle(radius number, area number);
declare
        x number;
        y number;
        pi constant number(4,2) := 3.14;
        area number(14,2);
begin
        x:= &x;
        y:= &y;
        while x<=y loop
        area := pi * power(x,2);
        insert into Circle(radius, area) values(x,area);
        x := x+1;
        end loop;
end;
/
```

2.

```
declare
cursor cur_circle is select * from Circle;
cursor_count cur_circle%rowtype;
totalrows number;
begin
open cur_circle;
loop
fetch cur_circle into cursor_count;
exit when cur_circle%notfound;
end loop;
totalrows := cur_circle%rowcount;
dbms_output.put_line('Total records in table Circle: '||totalrows);
end;
/
```

3.

```
prompt Enter a number
accept n
declare
num int;
digit int := 0;
```

```
ans int := 0;
begin
num := &n;
while num>0 loop
digit := num mod 10;
ans := ans*10;
ans := ans+digit;
num := floor(num/10);
end loop;
dbms_output.put_line('Reversed number: '||ans);
end;
/
```

4.

```
prompt Enter a number
accept num
declare
n int;
result int := 0;
digit int;
begin
n := &num;
while n>0 loop
digit := n mod 10;
result := result+digit;
n := floor(n/10);
end loop;
dbms_output.put_line('Sum of all digits of input number is: '||result);
end;
/
```

5.

```
create or replace function FindLargest (
num1 number,
num2 number,
num3 number
)return number is largest number;
begin
if (num1 >= num2) and (num1>=num3) then
largest := num1;
elsif (num1 <= num2) and (num2>=num3) then
largest := num2;
else
largest := num3;
```

```
end if;
return largest;
end FindLargest;
/

prompt Enter 1st num
accept x
prompt Enter 2nd num
accept y
prompt Enter 3rd num
accept z

declare
res number;
x number;
y number;
z number;
begin
x := &x;
y := &y;
z := &z;
res := FindLargest(x,y,z);
dbms_output.put_line('Largest number is: '||res);
end;
/
```

6.

```
create table Fact_table(n number,fact_n number);

create or replace function numFactorial(n number) return number is
fact number := 1;
i number;
begin
if n=0 then
fact := 1;
elsif n<0 then
RAISE_APPLICATION_ERROR(-20001, 'Factorial is not defined for negative numbers.');
else
for i in 1..n loop
fact := i*fact;
end loop;
end if;
insert into Fact_table values(n,fact);
return fact;
```

```
end numFactorial;
/

prompt Enter a number
accept num

declare
num number;
res number;
begin
num := &num;
res := numFactorial(num);
dbms_output.put_line(res);
end;
/
```

7. doubt in qn

8.

```
create or replace procedure dispFibonacci(n NUMBER) IS
a NUMBER := 0;
b NUMBER := 1;
i NUMBER := 0;
temp NUMBER := 0;
BEGIN
IF n <= 0 THEN
DBMS_OUTPUT.PUT_LINE('Invalid input. Please enter a positive integer.');
ELSE
DBMS_OUTPUT.PUT_LINE('The first ' || n || ' Fibonacci numbers are:');
DBMS_OUTPUT.PUT_LINE(a);
DBMS_OUTPUT.PUT_LINE(b);
IF n > 3 THEN
FOR i IN 3..n LOOP
a := a + b;
DBMS_OUTPUT.PUT_LINE(a);
temp := b;
b := a;
a := temp;
END LOOP;
END IF;
END IF;
END dispFibonacci;
/
```

```
prompt Enter number to find Fibonacci sequence:
accept r

declare
range number;
begin
range := &r;
dispFibonacci(range);
end;
/
```

9.

```
prompt Enter a number
accept n

declare
num number;
digit number := 0;
digit_count number := 0;
sum_result number := 0;
is_sum_prime number := 0;
i number;

begin
num := &n;
while num>0 loop
digit_count := digit_count+1;
digit := num mod 10;
sum_result := sum_result+digit;
num := floor(num/10);
end loop;
dbms_output.put_line('Total no of digits in input number is: '||digit_count);
dbms_output.put_line('Sum of all digits of input number is: '||sum_result);
for i in 2..(sum_result**0.5)+1 loop
if sum_result mod i = 0 then
dbms_output.put_line('Not Prime');
return;
end if;
end loop;
dbms_output.put_line('Prime');
end;
/
```

10.

```
prompt Enter number to check if Armstrong number or not
accept n

DECLARE
number_to_check NUMBER;
original_number NUMBER;
num_digits NUMBER := 0;
digit_sum NUMBER := 0;
BEGIN
   number_to_check := &n;
   original_number := number_to_check;
   WHILE number_to_check > 0 LOOP
      num_digits := num_digits + 1;
      number_to_check := floor(number_to_check / 10);
   END LOOP;

   number_to_check := original_number;

   WHILE number_to_check > 0 LOOP
      digit_sum := digit_sum+POWER(number_to_check MOD 10, num_digits);
      number_to_check := floor(number_to_check / 10);
   END LOOP;

   IF digit_sum = original_number THEN
      DBMS_OUTPUT.PUT_LINE(original_number || ' is an Armstrong number.');
   ELSE
      DBMS_OUTPUT.PUT_LINE(original_number || ' is not an Armstrong number.');
   END IF;
END;
/

11.
prompt Enter fullname
accept n
declare
full_name VARCHAR2(100);
formatted_name VARCHAR2(100);
first_name VARCHAR2(50);
last_name VARCHAR2(50);
begin
full_name := &n;
first_name := INITCAP(SUBSTR(full_name, 1, INSTR(full_name, ' ') - 1));
last_name := INITCAP(SUBSTR(full_name, INSTR(full_name, ' ') + 1));
```

```
formatted_name := SUBSTR(first_name, 1, 1) || '. ' ||last_name;

DBMS_OUTPUT.PUT_LINE('Original Name: ' || full_name);
DBMS_OUTPUT.PUT_LINE('Formatted Name: ' || formatted_name);
end;
/
```