Task 1

Write a function called **even_checker** that takes a number as an argument and prints whether the number is even or odd **inside the function**.

==========

Example1:
Function Call:
even_checker(5)
Output:
Odd!!

==========

Example2:
Function Call:
even_checker(2)
Output:
Even!!

Task 2

Write a python function that takes the limit as an argument of the Fibonacci series and prints till that limit.

==========

Function Call:
fibonacci(10)
Output:
0 1 1 2 3 5 8

==========

Function Call:
fibonacci(5)
Output:
0 1 1 2 3 5

Task 3

Write a function called **foo_moo** that takes a number as an argument and **returns** the following statements according to the below mentioned conditions. Then, finally prints the statement in the function call.

- If the number is divisible by 2, it should return "Foo".
- If the number is divisible by 3, it should return "Moo".
- If the number is divisible by both 2 and 3, it should return "FooMoo".
- Otherwise, it returns "Boo".

==========

Example1:
Function Call:
foo_moo(5)
Output:
Boo

Example2:
Function Call:
foo_moo(4)
Output:
Foo

Example3:
Function Call:
foo_moo(6)
Output:
FooMoo

Task 4

Write a python function that takes a string as an argument. Your task is to calculate the number of uppercase letters and lowercase letters and print them in the function.

==========

Function Call:
function_name('The quick Sand Man')
Output:
No. of Uppercase characters : 3
No. of Lowercase Characters: 12
==========
Function Call:
function_name('HaRRy PotteR')

Output:
No. of Uppercase characters : 5
No. of Lowercase Characters: 6

Task 5

Write a function called **calculate_tax** that takes 3 arguments: your age, salary, and current job designation.

Your first task is to take these arguments as user input and pass these values to the function.

Your second task is to implement the function and calculate the tax as the following conditions:

- **NO TAX IF YOU ARE LESS THAN 18 YEARS OLD.**
- **NO TAX IF YOU ARE THE PRESIDENT OF THE COMPANY**
- No tax if you get paid less than 10,000
- 5% tax if you get paid between 10K and 20K
- 10% tax if you get paid more than 20K

Finally return this tax value. Then print the returned value in the function call.

==========

Hints:
Here the job designation is a string, so it can be written in both uppercase and lower cases. So, you need to check the value ignoring the case.

==========

Example1:
Input:
16
20000
Student
Function Call:
calculate_tax(16, 20000, 'Student')
Output:
0

==========

Example2:
Input:
20
18000
assistant manager
Function Call:
calculate_tax(20, 18000, 'assistant manager')
Output:
900.0

==========

Example3:
Input:
20
22000
assistant manager
Function Call:
calculate_tax(20, 22000, 'Assistant manager')
Output:
2200.0

==========

Example4:
Input:
20
122000
president
Function Call:
calculate_tax(20, 122000, 'president')
Output:
0

Task 6

Write a function which will take 1 argument, number of days.

Your first task is to take the number of days as user input and pass the value to the function.

Your second task is to implement the function and calculate the total number of years, number of months, and the remaining number of days as output. No need to return any value, print inside the function.

Note: Assume, each year to be 365 days and month to be 30 days.

==========

Hint(1): Divide and mod the main input to get the desired output.

Hint(2): This task's calculation is similar to Assignment-1's seconds to hours, minutes conversion.

==========

Example01

Input:
4330
Function Call:
function_name(4330)
Output:
11 years, 10 months and 15 days

==========

Example02

Input:
2250
Function Call:
function_name(2250)
Output:
6 years, 2 months and 0 days

Task 7

Write a function called **show_palindrome** that takes a number as an argument and then returns a palindrome string. Finally, prints the returned value in the function call.

==========

Example1:
Function Call:
show_palindrome(5)
Output:
123454321

==========

Example2:
Function Call:
show_palindrome(3)
Output:
12321

Task 8

Write a function called **show_palindromic_triangle** that takes a number as an argument and prints a Palindromic Triangle in the function.

[Must reuse the show_palindrome() function of the previous task]

==========

Hints(1):
Need to use both print() and print(, end = " ") functions

==========

Example1:
Function Call:
show_palindromic_triangle(5)
Output:

```
        1
      1 2 1
    1 2 3 2 1
  1 2 3 4 3 2 1
1 2 3 4 5 4 3 2 1
```

==========

Example2:
Function Call:
show_palindromic_triangle(3)
Output:

```
    1
  1 2 1
1 2 3 2 1
```

Task 9

Write a function called **area_circumference_generator** that takes a radius of a circle as a function parameter and calculates its circumference and area. Then returns these two results as a **tuple** and prints the results using tuple unpacking in the function call according to the given format.

[Must use tuple packing & unpacking]

==========

Example1:
Function Call:
area_circumference_generator(1)
Output:
(3.141592653589793, 6.283185307179586)
Area of the circle is 3.141592653589793 and circumference is 6.283185307179586

==========

Example2:
Function Call:
area_circumference_generator(1.5)
Output:
(7.0685834705770345, 9.42477796076938)
Area of the circle is 7.0685834705770345 and circumference is 9.42477796076938

==========

Example3:
Function Call:
area_circumference_generator(2.5)
Output:
(19.634954084936208, 15.707963267948966)
Area of the circle is 19.634954084936208 and circumference is 15.707963267948966

Task 10

Write a function called **make_square** that takes a tuple in the parameter as a range of numbers (starting point and ending point (included)). The function should **return a dictionary** with the numbers as keys and its squares as values.

==========

Hints:
You need to declare a dictionary to store the result. You should use the range function to run the "for loop".

==========

Example1:
Function Call:
make_square((1,3))
Output:
{1: 1, 2: 4, 3: 9}

==========

Example2:
Function Call:
make_square((5,9))
Output:
{5: 25, 6: 36, 7: 49, 8: 64, 9: 81}

Task 11

Write a function called **rem_duplicate** that takes a tuple in the parameter and **return a tuple** removing all the duplicate values. Then print the returned tuple in the function call.

[Cannot use remove() or removed() for this task]

==========

Hints:
Unlike lists, tuples are immutable, so the tuple taken as an argument cannot be modified. But the list can be modified and lastly for returning the result use type conversion. You need to use membership operators (in, not in) for preventing adding any duplicates values.

==========

Example1:
Function Call:
rem_duplicate((1,1,1,2,3,4,5,6,6,6,6,4,0,0,0))
Output:
(1, 2, 3, 4, 5, 6, 0)

==========

Example2:
Function Call:
rem_duplicate(("Hi", 1, 2, 3, 3, "Hi",'a', 'a', [1,2]))
Output:
('Hi', 1, 2, 3, 'a', [1, 2])

Task 12

Write a python function that takes a list as an argument. Your task is to create a new list where **each element can be present at max 2 times**. Inside the function, print the number of elements removed from the given list. Finally, return the new list and print the result.

==========

Hint: You may use list_name.count(element) to count the total number of times an element is in a list. list_name is your new list for this problem.

==========

Function Call:
function_name([1, 2, 3, 3, 3, 3, 4, 5, 8, 8])
Output:
Removed: 2
[1, 2, 3, 3, 4, 5, 8, 8]
==========
Function Call:
function_name([10, 10, 15, 15, 20])
Output:
Removed: 0
[10, 10, 15, 15, 20]

Task 13

Write a python function that will perform the basic calculation (addition, subtraction, multiplication and division) based on 3 arguments. They are:

- Operator ('+', '-', '/', '*')
- First Operand (any number)
- Second Operand (any number)

Your first task is to take these arguments as user input and pass the values to the function parameters.

Your second task is to write a function and perform the calculation based on the given operator. Then, finally return the result in the function call and print the result.

==========

Input:
"+"
10
20
Function Call:
function_name("+", 10, 20)
Output:
30.0

==========

Input:
"*"
5.5
2.5
Function Call:
function_name("*", 5.5, 2.5)
Output:
13.75

Task 14

Write a function which will take 2 arguments. They are:

- Sentence
- Position

Your first task is to take these arguments as user input and pass these values to the function parameters.

Your second task is to implement the function and remove the characters at the index number which is divisible by the position (Avoid the index number 0 as it will always be divisible by the position, so no need to remove the index 0 character). Finally, add the removed characters at the end of the new string.

Return the value and then finally, print the new string at the function call.

[Cannot use remove() or removed() for this task]

==========

Input:
"I love programming."
3
Function call:
function_name("I love programming.", 3)
Output:
I lveprgrmmngo oai.

==========

Input:
"Python is easy to learn. I love python."
6
Function call:
function_name("Python is easy to learn. I love python.", 6)
Output:
Pythonis eay to earn.I lov pythn. sl eo

Task 15

You have been hired as an app developer for the company. The company plans to make an app for a grocery store where the user can order groceries and see the total amount to be paid in the cart section.

To build this feature, you have to write a function that takes 2 arguments. They are:

- order_items (must be a list)
- location (default value should be set to "Dhanmondi")

Your first task is to take a list of items from the user. Pass the list into the function parameter along with the optional location (Use default argument technique). (Also, no need to take location as input, pass this any value you want.)

Your second task is to implement the function. In the function, create a dictionary for the items shown in the table. Calculate the total price of the items passed as a list to the function. Additionally, add a delivery fee of 30 taka if the location is Dhanmondi. Otherwise, add a delivery fee of 70 taka. Finally, return the value and print it.

Item	**Price(Tk)**
Rice	105
Potato	20
Chicken	250
Beef	510
Oil	85

==========

Hint: The keys are the items and values are the corresponding price. Iterate the items in the list and check if the items in the list are available in the dictionary keys or not. If it is available, add the price.

==========

Example1:
function_name(["Rice", "Beef", "Rice"], "Mohakhali")

total = 105 + 510 + 105 = 720 (Take the price of each item and add them.)
total = 720 + 70 = 790 (Finally, add the delivery fee based on the location.)

Input:
["Rice", "Beef", "Rice"]
Function Call:
function_name(["Rice", "Beef", "Rice"], "Mohakhali")
Output:
790

==========

Example2:

function_name(["Rice", "Beef", "Rice"])

total = 105 + 510 + 105 = 720 (Take the price of each item and add them.)
total = 720 + 30 = 750 (Since no location is passed in the parameter, it will use the default location-"Dhanmondi". For Dhanmondi, delivery fee of 30 taka)

Input:
["Rice", "Beef", "Rice"]
Function Call:
function_name(["Rice", "Beef", "Rice"])
Output:
750

Optional Tasks (16 -21) [Ungraded]

Task 16

Write a function called **splitting_money** that takes an "amount" of money as an argument.
Your first task is to take the "amount" of money as user input and pass the value to the function parameter.
Your second task is to implement the function and calculate how that money can be split into 500, 100, 50, 20, 10, 5, 2, and 1 taka notes.
Then print the returned value in the function call.
==========

Hints:
This task's calculation is similar to Assignment-1's seconds to hours, minutes conversion. To return the result containing multiple strings, you need to store it in a variable and return it at the end of the function.
==========

Example1:
If the money is 1234, then the function should return
"500 Taka: 2 note(s)
100 Taka: 2 note(s)
20 Taka: 1 note(s)
10 Taka: 1 note(s)
2 Taka: 2 note(s)"
==========

Example2:
If the money is 151, then the function should return
"100 Taka: 1 note(s)
50 Taka: 1 note(s)
1 Taka: 1 note(s)"

Task 17

Write a function called **remove_odd** that takes a list of numbers that have both even and odd numbers mixed.
Your function should remove all the odd numbers and **return a compact list** which only contains the even numbers.
[Cannot use remove() or removed() for this task]
==========

Example1:
Function Call:
remove_odd ([21, 33, 44, 66, 11, 1, 88, 45, 10, 9])
Output:
[44, 66, 88, 10]
==========

Example2:
Function Call:
remove_odd ([11,2,3,4,5,2,0,5,3])
Output:
[2, 4, 2, 0]

Task 18

Write a function which will take 4 arguments. They are:

- starting value(inclusive)
- ending value(exclusive)
- first divisor
- second divisor

Your first task is to take these arguments as user input and pass these values to the function.

Your second task is to implement the function and find all the numbers that are divisible by the **first divisor or second divisor but not both** from the starting value(inclusive) and ending value(exclusive). Add all the numbers that are divisible and finally return this value. Print the returned value in the function call.

==========

Input:
10
40
4
7

Function Call:
function_name(10, 40, 4, 7)
Output:
210
==========
Input:
5
100
3
4
Function Call:
function_name(5, 100, 3, 4)
Output:
2012

Task 19

Write a python function which will take a string as an argument.

Your first task is to take a string as user input and pass the value to the function.

Your second task is to implement a function which will check whether all the alphabets from a to j (convert all the alphabets to lowercase) have appeared at least once in the given string or not.

- If all of these alphabets (a to j) appear at least once, then the result will be 5.
- If any one of the alphabets (a to j) is not in the given string, then the result will be 6.

Return this result and print the statement, "PSG will win the Champions League this season" that many times.

==========

Example01: "A black jackal is hunting a full grown deer"

Here all the alphabets from A to J are present at least once. So, the function will return 5 and will print the statement 5 times.

Input:
"A black jackal is hunting a full grown deer"
Function Call:
function_name("A black jackal is hunting a full grown deer")
Output:
PSG will win the Champions League this season
PSG will win the Champions League this season
PSG will win the Champions League this season

PSG will win the Champions League this season
PSG will win the Champions League this season

==========

Example02:

Input:
"ABBCDEFEFGHI"
Function Call:
function_name("ABBCDEFEFGHI")
Output:
PSG will win the Champions League this season
PSG will win the Champions League this season
PSG will win the Champions League this season
PSG will win the Champions League this season
PSG will win the Champions League this season
PSG will win the Champions League this season

Task 20

Write a function called **individul_bonus_calculation** which will take 4 arguments. They are:

- The player name
- Yearly earning of that player
- The total goal scored this season by that player
- Bonus percent per goal.

Your task is to implement the above-mentioned function that will calculate the total bonus on the yearly earnings of a player for the total goals he has scored.

Additionally,

- If the goal scored is above 30, add a (additional) bonus of 10000 taka.
- If it is between 20 and 30 inclusive, add an extra 5000 taka.

[For this task, there is no need to take any input from the user. Call the functions and print the values inside the function.]

==========

Example1:
individul_bonus_calculation("Neymar", 1200000, 35, 5)

bonus = 35 * (5 / 100 * 1200000) + 10000 = 2110000

Function call:
individul_bonus_calculation("Neymar", 1200000, 35, 5)
Output:
Neymar earned a bonus of 2110000 Taka for 35 goals.

==========

Example2:
individul_bonus_calculation('Jamal', 700000, 19, 8)

bonus = 19 * (8 / 100 * 700000) + 0 = 1064000

Function call:
individul_bonus_calculation('Jamal', 700000, 19, 8)
Output:
Jamal earned a bonus of 1064000 Taka for 19 goals.

==========

Example3:
individul_bonus_calculation('Luis', 80000, 25, 10)

bonus = 25 * (10 / 100 * 80000) + 5000 = 205000

Function call:
individul_bonus_calculation('Luis', 80000, 25, 10)
Output:
Luis earned a bonus of 205000 Taka for 25 goals.

==========

Task 21

You have been hired by the Abahani football club to write a function that will calculate the total bonus on the yearly earnings of each player for the total goals they have scored.

Since the number of players will vary, you decide to use the "*args" technique that you learned in your CSE110 class.

For each player: pass the name, yearly earning, the total goal scored this season, bonus percent per goal.

Additionally,

- If the goal scored is above 30, add an extra bonus of 10000 taka.
- If it is between 20 and 30 inclusive, add an extra 5000 taka.

[For this task, there is no need to take any input from the user. Call the functions and print the values inside the function.]

[Must reuse the individul_bonus_calculation() function of the previous task]

==========

Example1:
cal_bonus("Neymar", 1200000, 35, 5)

bonus = 35 * (5 / 100 * 1200000) + 10000 = 2110000

Function call:
cal_bonus("Neymar", 1200000, 35, 5)
Output:
Neymar earned a bonus of 2110000 Taka for 35 goals.

==========

Example2:
Function call:
function_name("Neymar", 1200000, 30, 10, "Jamal", 700000, 19, 5)
Output:
Neymar earned a bonus of 3605000 Taka for 30 goals.
Jamal earned a bonus of 665000 Taka for 19 goals.

==========

Example3:
Function call:
function_name("Neymar", 1200000, 35, 5, 'Jamal', 700000, 19, 8, 'Luis', 80000, 25, 10))
Output:
Neymar earned a bonus of 2110000 Taka for 35 goals.
Jamal earned a bonus of 1064000 Taka for 19 goals.
Luis earned a bonus of 205000 Taka for 25 goals.