CHAPTER -1

FUNCTIONS

Choose the best answer (1 Mark)

1. The small sections of code that are used to perform a particular task is called
   (A) Subroutines (B) Files (C) Pseudo code (D) Modules
2. Which of the following is a unit of code that is often defined within a greater code structure?
   (A) Subroutines **(B) Function** (C) Files (D) Modules
3. Which of the following is a distinct syntactic block?
   (A) Subroutines (B) Function **(C) Definition** (D) Modules
4. The variables in a function definition are called as
   (A) Subroutines (B) Function (C) Definition **(D) Parameters**
5. The values which are passed to a function definition are called
   (A) Arguments (B) Subroutines (C) Function (D) Definition
6. Which of the following are mandatory to write the type annotations in the function definition?
   (A) Curly braces **(B) Parentheses** (C) Square brackets (D) indentations
7. Which of the following defines what an object can do?
   (A) Operating System (B) Compiler **(C) Interface** (D) Interpreter
8. Which of the following carries out the instructions defined in the interface?
   (A) Operating System (B) Compiler **(C) Implementation** (D) Interpreter
9. The functions which will give exact result when same arguments are passed are called
   (A) Impure functions (B) Partial Functions (C) Dynamic Functions (**D) Pure functions**

10. The functions which cause side effects to the arguments passed are called
    (A) impure function (B) Partial Functions
    (C) Dynamic Functions (D) Pure functions

A.Baseera Nashrin M.sc., B.Ed.,

Part - II

Answer the following questions (2 Marks)

1. What is a subroutine?

Subroutines are small sections of code that are used to perform a particular task that can be used repeatedly.

2. Define Function with respect to Programming language.

A function is a unit of code that is often defined within a greater code structure .A function contains a set of code that works on many kinds of inputs and produces a concrete output.

3. Write the inference you get from X:=(78).

The value of 78 bound to the name 'X'.

4. Differentiate interface and implementation.

Interface	Implementation
Interface just defines what an object can do, but won't actually do it.	Implementation carries out the instructions defined in the interface .

5. Which of the following is a normal function definition and which is recursive function definition

i) let rec sum x y:
return x + y
ii) let disp :
print 'welcome'
iii) let rec sum num:
if (num!=0) then return num + sum (num-1)
else
return num

Answer:

(i) Recursive Function

(ii) Normal Function

(iii) Recursive Function

Part - III

Answer the following questions (3 Marks)

1. Mention the characteristics of Interface.

- The class template specifies the interfaces to enable an object to be created and operated properly.
- An object's attributes and behaviour is controlled by sending functions to the object

2. Why strlen is called pure function?

strlen is a pure function because the function takes one variable as a parameter, and accesses it to find its length. This function reads external memory but does not change it, and the value returned derives from the external memory accessed.

3. What is the side effect of impure function. Give example.

The most popular groups of side effects is modifying the variable outside of function.

Example:

```
let y: = 0
(int) inc (int) x
y: = y + x;
```

```
return (y)
```

In the above example the value of y get changed inside the function definition due to which the result will change each time. The side effect of the inc () function is it is changing the data of the external visible variable ***'y'***.

4. Differentiate pure and impure function.

Pure Function	Impure Function
The return value of the pure functions solely depends on its arguments passed. Hence, if you call the pure functions with the same set of arguments, you will always get the same return values. They do not have any side effects.	The return value of the impure functions does not solely depend on its arguments passed. Hence, if you call the impure functions with the same set of arguments, you might get the different return values For example, random(), Date().
They do not modify the arguments which are passed to them	They may modify the arguments which are passed to them

5. Wha happens if you modify a variable outside the function? Give an example.

One of the most popular groups of side effects is modifying the variable outside of function.

Example

```
let y: = 0
(int) inc (int) x
y: = y + x;
return (y)
```

Example Explanation:

In the above example the value of y get changed inside the function definition due to which the result will change each time. The side effect of the inc () function is it is changing the data of the external visible variable 'y'. A good sign that our function impure (has side effect) is that it doesn't take any arguments and it doesn't return any value.

Part - IV

Answer the following questions (5Marks)

1. What are called Parameters and write a note on
(i) Parameter without Type (ii) Parameter with Type

Parameters:

Parameters are the variables in a function definition and arguments are the values which are passed to a function definition through the function definition.

(i) Parameter without Type

```
(requires: b>=0 )
(returns: a to the power of b)
let rec pow a b:=
if b=0 then 1
else a * pow a (b-1)
```

In the above function definition **variable *'b'* is the parameter** and the value which is passed to the variable *'b'* is the argument. The precondition ***(requires)*** and post condition ***(returns)*** of the function is given. Note we have not mentioned any types ***(data types)***.

(ii) Parameter with Type

```
(requires: b> 0 )
(returns: a to the power of b )
let rec pow (a: int) (b: int) : int :=
if b=0 then 1
else a * pow b (a-1)
```

When we write the type annotations for 'a' and 'b' the parentheses are mandatory. This is called parameter with type.

2. Identify in the following program

let rec gcd a b :=
if b <> 0 then gcd b (a mod b) else return a

i) Name of the function

ii) Identify the statement which tells it is a recursive function

iii) Name of the argument variable

iv) Statement which invoke the function recursively

v) Statement which terminates the recursion

(i) gcd
(ii) rec
(iii) b, a mod b
(iv) gcd b (a mod b)
v) return a

3. Explain with example Pure and impure functions.

Pure functions

Pure functions are functions which will give exact result when the same arguments are passed.

Example:

```
let square x
return: x * x
```

The above function square is a pure function because it will not give different results for same input.

Impure functions

The variables used inside the function may cause side effects though the functions which are not passed with any arguments. In such cases the function is called impure function.

Example:

```
let Random number
let a := random()
if a > 10 then
```

```
return: a
else
return: 10
```

When a function depends on variables or functions outside of its definition block, we can never e sure that the function will behave the same every time it's called. For example the mathematical function random() will give different outputs for the same function call.

4. Explain with an example interface and implementation.

Interface:

- An interface is a set of action that an object can do .Interface just defines what an object can do, but won't actually do it .
- In object oriented programs classes are the interface .

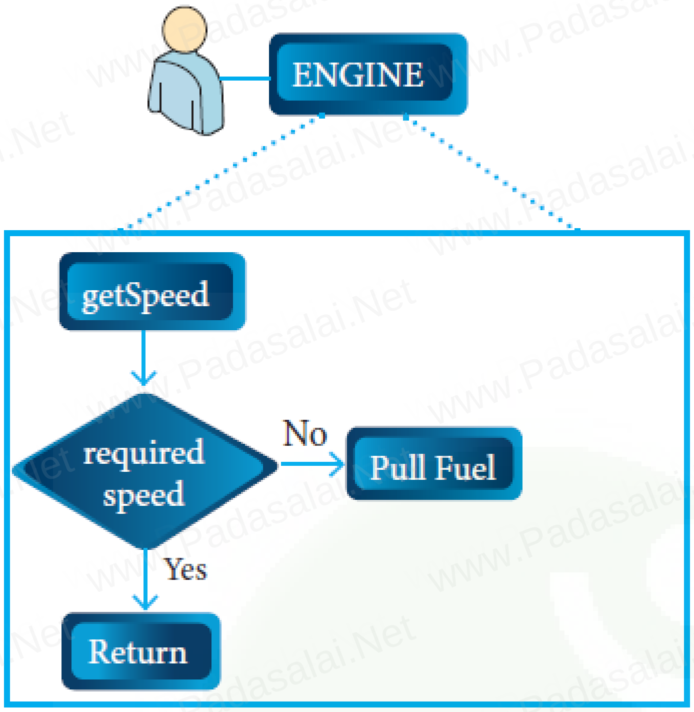
Characteristics of interface

- The class template specifies the interfaces to enable an object to be created and operated properly.
- An object's attributes and behaviour is controlled by sending functions to the object.

Implementation:

- Implementation carries out the instructions defined in the interface.
- In object oriented programs the object is processed and executed is the implementation.

Example:



The person who drives the car doesn't care about the internal working. To increase the speed of the car he just presses the accelerator to get the desired behaviour. Here the accelerator is the interface between the driver *(the calling / invoking object)* and the engine *(the called object)*.

In this case, the function call would be Speed (70): This is the interface.

Internally, the engine of the car is doing all the things. It's where fuel, air, pressure, and electricity come together to create the power to move the vehicle. All of these actions are separated from the driver, who just wants to go faster. Thus we separate interface from implementation.

A.BASEERA NASHRIN , M.Sc., B.Ed.,

nashrinbaseera@gmail.com

8300980856

****