Visit QB365-SLMS - https://www.qb365.in/slms/ Complete Questions and Answers

Answers Key:

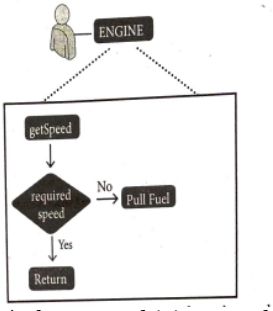
IV. ANSWER ALL THE QUESTIONS.

1. **Parameters and arguments:** Parameters are the variables used in a function definition. Arguments are the values passed to those parameters when the function is called.
   (i) Parameter without Type: Let us see an example of a function, definition :
   (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 postcondition (returns) of the function is given.
   • In this function, we have not mentioned any data types.
   • Some compilers can infer types automatically, while others require the programmer to specify them.
   • In the above example:
   • The if expression returns 1 an integer), so the whole expression has type int.
   • Therefore, the function's return type is int.
   • Since b is compared with 0, it must be of type int.
   • Because a is multiplied using *, it must also be of type int.
   (ii) Parameter with Type: Now, let's define the same function with types:
   (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 a (b-1)
   • When writing type annotations for 'a' and 'b', parentheses are mandatory.
   • Usually, we can omit type annotations because the compiler can infer them.
   • However, adding types explicitly can help debug type errors more easily.
   • The keyword let introduces a function definition.
   • To define a recursive function, use let rec instead of just let.
   Syntax:
   let rec fn al a2 ... an := k
   • fn - function name
   • a1 to an - parameters
   • rec - required if the function calls itself (recursive function)
   • Here the fn is used as a function name. The names 'al' to 'an' are variables used as parameters. The keyword rec' is required if 'fn' is to be a recursive function; otherwise it may be omitted.
2. **Pure functions:**
   (i) Pure functions give exact results when the same arguments are passed.
   (ii) Example: The mathematical function sin(0) always results in 0.
   (iii) A pure function should not have any external variable that alters its behavior.
   (iv) Example :
   let square x :=
   return: x* x
   Here, square is pure because it gives the same result for the same input.
   (v) One advantage is that if a function is pure, then if it is called several times with the same arguments, the compiler only needs to actually call the function once..
   Impure functions :
   (i) 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
   (ii) When a function depends on variables or functions outside of its definition block, you can never be sure that the function will behave the same every time it's called.
   (iii) Example :
   let randomnumber:=
   a := random()
   if a > 10 then
   return: a
   else
   return: 10
   (iv) Here, random() is impure because it gives different results for the same function call.
3. **Interface :**

(i) An interface is a set of action that an object can do. For example when you press a light Switch, the light goes on, you may not have cared how it splashed the light. In Object Oriented Programming language, an Interface is a description of all functions.
(ii) In our example, anything that "ACTS LIKE" a light, should have function definitions like turn_on () and a turn off (). The purpose of interfaces is to allow the computer to enforce the properties of the class.

Implementation :

(i) Implementation carries out the instructions defined in the interface.
(ii) How the object is processed and executed is the implementation
For example, let's take the example of increasing a car's speed



(iv) The person driving a car doesn't need to know how the engine works. To increase speed, the driver simply presses the accelerator to get the desired result. Here, the accelerator acts as the interface between the driver (the calling/invoking object) and the engine (the called object).
(v) For example, the function call could be Speed(70):This is the interface.
(vi) Inside the engine, fuel, air, pressure, and electricity work together to create power. These internal actions are hidden from the driver, who only wants to go faster.
(vii) Thus, the interface (accelerator) is separated from the implementation (engine working).

III. ANSWER ANY SIX QUESTIONS.

4. (requires: x>= 0)
let rec even x:=
x=0 || odd (x-1)
and odd x:=
X<>0 && even (x-1)
The value of the expression (T/F) is automatically the return value of the function, the last expression evaluates to a boolean and is returned automatically.
5. (i) The syntax to define functions is close to the mathematical usage: the definition is introduced by the keyword let, followed by the name of the function and its arguments; then the formula that computes the image of the argument is written after an := sign. If you want to define a recursive function: use "let rec" instead of "let".
(ii) Syntax : The syntax for function definitions:
let rec fn a1 a2 ... an : = k
(iii) Here the 'fn' is used as a function name. The names 'al' to an' are variables used as parameters. The keyword 'ree' is required if 'fn' is to be a recursive function; otherwise it may be omitted.
6. (i) The class template specifies the interfaces to enable an object to be created and operated properly.
(ii) An object's attributes and behaviour is controlled by sending functions to the object.
7. (i) strlen is a pure function because the function takes one variable as a parameter, and accesses it to find its length.
(ii) This function reads external memory but does not change it, and the value returned derives from the external memory accessed.
8. 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. When a function depends on variables or functions outside of its definition block, you can never be sure that the function will behave the same every time it's called.
Example random( ) function.
9.

S. No	Pure	Impure
(i)	The return value of the pure functions solely depends on its arguments passed.	The return value of the impure functions does not solely depend on its arguments passed.
(ii)	If you call the pure functions with the same set of arguments, you will always get the same return values.	If you call the impure functions with the same set of arguments, you might get the different return values.
(iii)	They do not have any side effects.	They have side effects For example: random(), Date().
(iv)	They do not modify the arguments which are passed to them	They may modify the arguments which are passed to them

QB365 Study Material QUESTION BANK 365

10. let min x y z :=
    if x < y then
    if x < z then x else z
    else
    if y < then y else z

II. ANSWER ANY SIX QUESTIONS.

11. Parameters are the variables in a function definition and arguments are the values which are passed to a function definition.
12. let rec monochromatize a b c : =
    if a > 0 then
    a, b, c := a-1, b-1,c+2
    else
    monochromatize abc
    return a, b, c
13. (i) Subroutines are the basic building blocks of computer programs. Subroutines are small sections of code that are used to perform a particular task that can be used repeatedly.
    (ii) In Programming languages these subroutines are called as Functions.
14. A function is a unit of code that is often defined within a greater code structure. Specifically, a function contains a set of code that works on many kinds of inputs, like variables, expressions and produces a concrete output
15. The difference between interface and implementation is

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

16. let rec sum num:
    if (num!=0) then return num + sum(num-1)
    else
    return num
17. Pure functions are functions which will give exact result when the same arguments are passed. For example the mathematical function sin (0) always results 0.

QB 365 TM Study Material
QUESTION BANK 365

I. CHOOSE THE CORRECT ANSWER:

18. (a) Subroutines
19. (a) Arguments
20. (d) Pure functions
21. (c) Subroutine
22. (b) random()