

## Sheet#2

الاسم/حسناء محمد احمد دياب خاطر

202000273

Sheet#2 حسناء محمد احمد دياب خاطر  
202000273 / 15

① List 4 different Paradigms and compare them?

| Functional                         | Logic       | Procedural              | Object-oriented                  |
|------------------------------------|-------------|-------------------------|----------------------------------|
| It's form of declarative           | declarative | imperative              | imperative                       |
| *examples:-                        | *examples:- | *examples:-             | *examples:-                      |
| • LISP • Haskell<br>• ML • Miranda | • Prolog    | • Algol<br>• C<br>• C++ | • Smalltalk<br>• Simula<br>• C++ |

② What is invariant programming and which Paradigms is helpful for them?

- is Program correct and efficient. Loops where recursive call is the last operation done in the function body, and it use Accumulator and Principle of communicating vases to achieve that.

- It applies to both declarative and imperative Paradigms.

③ declare

```

fun fact N
  if N == 0 then 1
  else
    N * fact N-1
  end
end
    
```

202000273 / 1d  
1 1

④ What is tail recursive and why it's important?

- it is the recursive call is the last operation done in the function body
- it improves loop, make loops correct and efficient.

⑤ declare

```
fun {PrintL from to}
  if from > to then nil
  else
    from 1 {PrintL from+1 to}
  end
end
```

⑥

declare

fun {Sumfac N A}

if  $N = 0$  then A

else

{Sumfac  $N-1$   $A*N$ }

end

end

2020/10/21  
202000273/10  
1 1

② declare  
Fun {checkPrime N m:3  
  if  $m == 1$  then true  
  else  
    if  $(N \bmod m) == 0$  then false  
    else  
      {checkPrime N m-1} end  
    end  
  end  
end  
Fun {isPrime N:  
  if  $N == 1$  then true  
  else  
    if  $N == 2$  then false  
    else {checkPrime N N-1} end  
  end end  
{BBrwse 73}

10/10/2020 18:10  
202000273/10  
/ /

⑨ both are recursive but tail recursion is the last operation done in the function body

⑩ declare  
fun {SumT N A}  
 if N == 0 then A  
 else  
 {SumT N-1 A+(N\*A)} end  
end  
{Browse {SumT 3 0}} → 14

⑩ declare  
fun {Print F T A}  
 if T < F then A  
 else  
 {Print F T-1 T+A} end  
end  
{Browse {Print 1 5 0}}

15  
16