1)
Paradigma de programação são padrões de programação suportado por diversas linguagens que agrupam certas características comuns. Existem 2 paradigmas, o imperativo, em que as sequencias de instruções que mudam as células da memória. O outro paradigma é o declarativo, em que o conjunto de definiçoes de funçoes que aplicamos a valores.
Exemplos:
BASIC, C, FORTRAN, Java, c++, c#, python, javascript - imperativo
Haskell, scheme, Clojure, Prolog, Curry - declarativo

2)
O computador sabe porque já está definido no módulo padrão da linguagem, o Prelude. Este módulo contém um grande conjunto de funçoes pré-definidas. Ele é carregado junto com interpretador/compilador e pode ser usado em qualquer programa haskell.

3)
O compilador(GHC), através do programa de alto nível, faz todas as análises sintáticas e demais processos e por fim gera um programa executável. Já o interpretador(GHCi), não ocorre a geraçao de nenhum novo arquivo, é feita uma traduçao instantanea em tempo de execução.

4)
A definiçao é feita automaticamente pela própria linguagem, essa característica é inerente ao Haskell e se chama deduçao automática de tipos(inferencia de tipos). A vantagem é que por ser uma linguagem fortemente tipada e então causa uma compilação mais eficiente do código.

5)
O tipo Char representa caracteres simples enquanto o tipo String representa uma sequencia de caracteres. As expressões 'a' e "a" representam valores diferentes, a primeira é entendida como um caracter simples, enquanto a segunda representa uma sequencia de caracteres(por mais que essa sequencia tenha apenas 1 caracter).

6)
Alternativas 'a' e 'e'. Nomes de funções começam com letra minúscula e podem incluir letras, digitos, sublinhados e apóstrofes.

7)
8)
9)
10)

```haskell
subtracao :: Double -> Double -> Double
subtracao x y = x - y

areaCirculo :: Double -> Double
areaCirculo raio = raio * raio * pi

diferencaAreaCirculo :: Double -> Double -> Double
diferencaAreaCirculo raio1 raio2 = subtracao (areaCirculo raio1) (areaCirculo raio2)

logica :: Bool -> Bool -> Bool
logica p q = (p || q) && not (p && q)

main :: IO ()
main = do
 print ("Subtracao")
 print (subtracao 6 2)
 print ("Area circulo")
 print (areaCirculo 6)
 print ("Diferenca Area circulo")
 print (diferencaAreaCirculo 8 6)
 print ("funcao logica")
 print (logica True False)
```