algobase-training-course

TEST 2 (après étape 4)

Dans la suite, on suppose que les deux lignes suivantes sont écrites au début des programmes

```
/// <reference path="fonctions.ts"/>
creerZoneDessin();
```

Analysez, comprenez le code avant de le tester / modifier !

Question 1

Soit le code suivant:

```
let unnombre:number=lireNombre("Entrez un nombre");
if(unnombre=10)
{
    afficherTexte("Le nombre est 10");
}
```

- Le message sera toujours affiché (même si on modifie la valeur affectée à unnombre). Pourquoi ? Corrigez

---

Question 2

Soit le code suivant:

```
let unnombre:number=lireNombre("Entrez un nombre");
if(unnombre>=10)
{
    afficherTexte("Le nombre est supérieur ou égal à 10");
}
else(unnombre<10)
{
    afficherTexte("Le nombre est infèrieur à 10");
}
```

- Corrigez ce code

Question 3

Soit le code suivant:

```
let unnombre:number=lireNombre("Entrez un nombre");
if(unnombre==10)
{
    afficherTexte("Le nombre vaut 10");
}
else if(unnombre==20)
{
    afficherTexte("Le nombre vaut 20");
}
afficherTexte("Le nombre est différent de 10 et 20");
```

- Pourquoi le texte "Le nombre est différent de 10 et 20" est-il toujours affiché, quel que soit la saisie de l'utilisateur ? Corrigez ce code

---

Question 4

On veut maintenant obliger l'utilisateur à saisir un nombre supérieur ou égal à 10:

```
let unnombre:number;
while(unnombre<10)
{
    lireNombre("Entrez un nombre supérieur ou égal à 10");
}
afficherTexte("Bravo, le nombre unnombre est bien plus grand que 10");
```

- Quelles sont les deux erreurs présentes dans ce code ? Corrigez-les et testez.

Question 5

On veut maintenant obliger l'utilisateur à saisir un nombre entre 10 et 20. Nous modifions la condition dans le TANT QUE (par rapport à votre code précédent, corrigé):

```
while( (unnombre<10) && (unombre>20))
```

i && est la traduction en TypeScript de l'opérateur logique ET

- Est-il possible que unnombre soit à la fois plus petit que 10 et plus grand que 20 ? Si ce n'est pas le cas, la condition dans le TANT QUE sera toujours fausse ... et donc on ne rentrera jamais dans la boucle (ie: le message final sera toujours affiché) => remplacez le ET par un OU (&& => ||)