

Informatique 2 : (11-12)

Examen (Durée 1h30)

Soit $T[N]$ un tableau d'entiers positifs. Et soit une séquence d'entiers (0..9) qui se termine par un nombre (supérieur à 9 ou inférieur à 0).

On remplit le tableau par cette séquence. La taille du tableau est supérieur au nombre de chiffres de cette séquence. Les chiffres du tableau représentent un entier positif.

- Exemple1 : Soit la séquence 1 4 3 12

Pour $N=5$ T

1	4	3	12	
---	---	---	----	--

Le nombre représenté par le tableau T est 143 et sa taille = 3.

- Exemple2 : Soit la séquence 45

Pour $N=5$ T

45				
----	--	--	--	--

Le nombre représenté par le tableau T n'existe pas et sa taille = 0.

- Exemple3 : Soit la séquence -3

Pour $N=5$ T

-3				
----	--	--	--	--

Le nombre représenté par le tableau T n'existe pas et sa taille = 0.

Ecrire les sous programmes de manipulation d'un tel tableau suivants :

- Remplissage du tableau T à partir de la séquence.(3pts)
- Renvoi de la taille du nombre représenté par le tableau.(2pts)
- Calcul de ce nombre et retourne ce nombre au programme appelant.(3pts)
- Calcul du nombre inverse sans l'utilisation de pile.(4pts)
- Calcul du nombre inverse en utilisant une pile qu'il faut déclarer ainsi que les sous programmes de manipulation de pile auxquels vous faites appel.(4pts)
- Renvoi du premier nombre premier supérieur à ce nombre.(4pts)

Informatique 2 : (11-12)

Corrigé de l'examen

```
#include<stdio.h>
```

```
const int N=10;
```

a) void remplissage(int T[N])

```
{int nb,i=0;
```

```
printf("Donner une valeur ");
```

```
scanf("%d",&nb);
```

```
while (nb>=0 && nb<=9)
```

```
{T[i]=nb;
```

```
i++;
```

```
printf("Donner une autre valeur ");
```

```
scanf("%d",&nb);
```

```
}
```

```
T[i]=nb;}
```

(3)

b) int taille(int T[N])

```
{int i=0;
```

```
while (T[i]>=0 && T[i]<=9)
```

```
i++;
```

```
return i;}
```

(2)

c) long nombre(int T[N])

```
{long nb=0;
```

```
int i=0;
```

```
while (T[i]>=0 && T[i]<=9)
```

```
{nb=nb*10 + T[i];
```

```
i++;}
```

```
return nb;}
```

(3)

d) long inverse1(long nb)

```
{long inv1 = 0;
```

```
while (nb!=0)
```

```
{inv1 = inv1*10 + nb%10;
```

```
nb=nb/10;}
```

```
return inv1;}
```

(4)

e) typedef struct

```
{int tab[N];
```

```
int nbval;
```

```
}pile;
```

(0,5)

```
void init(pile *p)
```

```
{p->nbval=0;}
```

(0,25)

```
int estvide(pile p)
```

```
{return (p.nbval==0)?1:0; }
```

(0,25)


```
void mettre(pile *p,int x)
{p->tab[p->nbval]=x;
 p->nbval++;}
```

0,5

```
int enlever(pile *p)
{int x;
 x=p->tab[p->nbval-1];
 p->nbval--;
 return x;}
```

0,5

```
long inverse2(int T[N])
{long inv2=0,x;
 pile p;
 int i=0;
 init(&p);
 while(T[i]>=0 && T[i]<=9)
 {mettre(&p,T[i]);
  i++;}
 while (!estvide(p))
 {x=enlever(&p);
  inv2 = inv2*10 + x;
 }
 return inv2;}
```

1

1

```
f) int premier(long nb)
{long i;
 for (i=2;i<=nb/2;i++)
  if (nb%i==0)
   return 0;
 return 1; }
```

2

```
long prochainpremier(long nb)
{long i;
 i=nb+1;
 while (!premier(i))
  i++;
 return i;}
```

2