

Examen d'informatique

Question 1 (2 points)

Écrire une fonction `int suite(int n)` qui calcule l'entier s_n défini par

- $s_0 = 2$,
- $s_1 = 4$,
- $s_n = 2s_{n-1} + 3s_{n-2}$ pour $n \geq 2$.

```
int suite (int n){ // n est supposé positif ou nul
    int u,v,w, i;
    if (n == 0) return 2;
    if (n == 1) return 4;
    else{
        u = 2; v = 4;
        for (i=0; i <= n-2; i++){
            w = u;
            u = v;
            v = 2*v + 3*w;
        }
        return v;
    }
}
```

On considère dans la suite des chaînes de caractères qu'on appellera aussi des mots. On utilisera seulement des mots formés de lettres minuscules. On utilisera les constantes `#define N 20` et `#define NB_LETTRES 26`.

Question 2 (3 points)

Écrire une fonction `void saisieControlee(char s[])` qui effectue la saisie au clavier d'une ligne d'entrée formée d'au plus $N-1$ lettres minuscules. Seules les minuscules seront mémorisées.

```

void saisieControlee(char s[]){
    int i = 0;
    char c;
    printf("Entrer un mot en lettres minuscules d'au plus %d caractères",N-1);
    do {
        scanf("%c",&c);
        if (('a'<= c) && (c <= 'z')) {s[i] = c; i++;}
    } while ((i < N-1) && (c != '\n'));
    s[i] = '\0';
}

```

Question 3 (2 points)

Écrire une fonction `int frequence(char c, char s[])` qui rend le nombre d'occurrences du caractère `c` dans la chaîne de caractères `s`.

```

int frequence(char c, char s[]){
    int i,r=0;
    for(i=0; s[i]!='\0'; i++)
        if (s[i]==c) r++;
    return r;
}

```

Question 4 (2 points)

Écrire une fonction `void tableauFrequences(char s[], int table[])` qui place dans le tableau `table` (de taille `NB_LETTRES`) les fréquences des différentes lettres qui composent la chaîne `s`.

```

void tableauFrequences(char s[], int table[]){
    int i;
    for(i=0;i<NB_LETTRES;i++)
        table[i]=0;
    for(i=0;s[i]!='\0';i++)
        table[s[i]-'a']++;
}

```

Question 5 (2 points)

Écrire la fonction `main` qui lit un mot formé d'au plus `N-1` caractères minuscules et affiche les lettres et leur fréquence dans le mot.

```

int main(void){
    char s[N];
    int frequence[NB_LETTRES];
    int i;
    saisieControlee(s);
    tableauFrequence(s, frequence);
    for (i=0; i< NB_LETTRES; i++)
        printf("%c : %d\n", 'a'+i, frequence[i]);
    return 0;
}

```

Question 6 (2 points)

Écrire une fonction `void contenu(int a[], int temoin[])` qui prend en argument deux tableaux `a` et `temoin` de taille `N`. Les valeurs de `a` sont comprises entre 0 et `N-1`. Elle remplit le tableau `temoin` de façon que `temoin[j]` est égal au nombre de fois que la valeur `j` apparaît dans le tableau `a`.

```

int contenu(int a[], int temoin[]){
    int i;
    for(i=0;i<N;i++) temoin[i]=0;
    for(i=0;i<N;i++) temoin[a[i]]++;
}

```

On appelle *permutation* un tableau de taille `N` contenant chaque entier `i` avec $0 \leq i < N$ une fois et une seule.

Question 7 (2 points)

Écrire une fonction `int estUnePermutation(int a[])` qui renvoie 1 si le tableau `a` de taille `N` est une permutation et 0 sinon.

```

int estUnePermutation(int a[]){
    int i;
    int b[N];
    contenu(a,b);
    for(i=0;i<N;i++){
        if(b[i] != 1) return 0;
    }
    return 1;
}

```

Question 8 (2 points)

L'inverse d'une permutation a est la permutation b telle que $b[a[i]] = i$ pour tout indice i . Écrire une fonction `void inverse(int a[], int b[])` qui calcule dans b l'inverse de la permutation a . Les tableaux a et b auront la taille N .

```
void inverse(int a[], int b[]){
    int i;
    for(i=0;i<N;i++) b[a[i]]=i;
}
```

Question 9 (3 points)

La *permutation standard* d'un mot de longueur N est obtenue de la façon suivante. On numérote en ordre croissant à partir de 0 les lettres a du mot, puis les lettres b , puis les lettres c , etc. La suite des numéros obtenus en enlevant les lettres forment la permutation standard. Ainsi la permutation standard de `baacdabba` est le tableau de taille 9 contenant 401782563.

```
b a a c d a b b a
  0 1      2      3
4          5 6
      7
      8
-----
4 0 1 7 8 2 5 6 3
```

Écrire une fonction `void permutationStandard(char s[], int p[])` qui calcule dans le tableau p de taille N la permutation standard d'un mot s de longueur N . (On pourra calculer un tableau indiquant, pour chaque lettre apparaissant dans s , le numéro de son occurrence la plus à gauche).

```
void permutationStandard(char s[], int p[]){
    int frequence[NB_LETTRE];
    int numero[NB_LETTRES];
    int i;
    tableauFrequences(s, frequence);
    numero[0]=0;
    for (i=1; i < NB_LETTRES; i++)
        numero[i]= numero[i-1] + frequence[i-1];
    for (i=0; i < s[i] != '\0; i++)
        {p[i] = numero[s[i]-'a']; numero[s[i]-'a']++;}
}
```