Université de Tlemcen
Faculté des Sciences
1re année Licence MI

Examen Final d'Algorithmique2

Année 2022-2023
Semestre 2
22/05/2023
Durée : 1h30

NOM : PRÉNOM :

DATE DE NAISSANCE :

Exercice 1 (5 pts) Cet exercice compte aussi pour des points bonus au contrôle

Le programme dont le code source est en page annexe1 vous permettra de faire réviser l'alphabet à votre petite sœur ou petit frère. Il lui manque juste les 10 petits morceaux que vous voyez sous le programme. Indiquez ci-dessous **le numéro de ligne** où il faut placer chaque morceau de code pour compléter le programme :

a :	b :	c :	d :	e :	f :	g :	h :	i :	j :

Questions diverses (11 pts)

1) Si t est défini par `int t[2][3]={{3,4,5}{0,1,2}} ;`

alors t[0][2] vaut : ☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ erreur

2) Qu'affichent les instructions suivantes ?

```
char s1[]="Caractere";
char *s2="Salutations!";
printf("%d %d %d %d", sizeof s1, strlen(s1), strlen(s2), (sizeof s2>sizeof s1));
```

☐ 9 9 12 1 ☐ 9 10 13 1 ☐ 10 9 12 0 ☐ 10 9 12 1 ☐ 9 9 12 0 ☐ aucune des 5

3) Observez les instructions ci-dessous :

```
struct objet{int x,y;};
.....
int b[3]={10,20,30};
struct objet w={10,20};
a=b; v=w;
a[1]=42; v.y=42;
printf("%d %d", b[1], w.y);
```

a) Avec quoi peut-on les compléter (2e ligne) pour qu'elles se compilent sans erreurs ? (cochez une seule réponse)

☐ `int a[3]; struct objet v;`
☐ `int a[]; struct objet *v;`
☐ `int *a; struct objet v;`
☐ `int a[3]; struct objet *v;`
☐ `int *a; struct objet *v;`

b) Qu'affichent-elles ? (une seule réponse)

☐ `42 42` ☐ `42 20` ☐ `0 20` ☐ `20 42` ☐ `20 20` ☐ `Aucune des 4`

4) L'appel à la fonction `fopen("mondossier/fichier.txt", "w");` permet d'ouvrir le fichier "fichier.txt" en écriture seule. Dans quels cas peut-il renvoyer la valeur NULL ? (cochez la ou les réponses justes)

☐ Si la taille du fichier est zéro.
☐ Si le fichier n'existe pas dans "mondossier".
☐ Si "mondossier" n'existe pas.
☐ Si le disque qui contient le fichier est endommagé.
☐ Aucun des 4 cas.

5) Quelles instructions sont équivalentes à l'instruction `puts(chaine);` Cochez la ou les réponses justes :

☐ `fputs(chaine, stdin);`
☐ `fputs(chaine, stdout);`
☐ `fputs(chaine, screen);`
☐ `fprintf(stdout, "%s\n",chaine);`
☐ `fprintf(stdin, "%s\n",chaine);`
☐ `fscanf(stdout, "%s\n", chaine);`
☐ `fscanf(stdin, "%s\n", chaine);`

6) Soit la déclaration suivante :

```
int a=42, *p=&a, *q=&p[0];
```

Que vaut l'expression `(*q)` ? (cocher une seule bonne réponse)

☐ L'adresse de p ☐ l'adresse de p[0] ☐ la valeur de a, c'est-à-dire 42
☐ *p, c'est-à-dire l'adresse de a ☐ aucune des 4

7) Soit la déclaration suivante :

```
int a[]={10,20,30}, *p=&a[0], *q=&p[0];
```

Que vaut l'expression `(*q)` ? (cocher une seule réponse)

☐ L'adresse de p ☐ l'adresse de p[0]
☐ la valeur de a[0], c'est-à-dire 10
☐ *p, c'est-à-dire l'adresse de a[0] ☐ aucune des 4

8) Soit la déclaration suivante :

```
int m[2][3]={{10,20,30},{40}};
```

Que vaut l'expression **(m+1) ?(cocher une seule réponse)

☐ 10 ☐ 20 ☐ 40
☐ 0 ☐ une valeur indéfinie ☐ aucune des 5 réponses

9) Dans le programme ci-contre : (<u>une seule</u> bonne réponse)

☐ la variable **a** est globale, **n** et **cpt** sont statiques, **b** est locale.
☐ **a**, **b**, **n** et **k** sont des variables automatiques et **cpt** est statique
☐ il y a 2 variables appelées **a**, l'une est statique, l'autre automatique
☐ **n** est un paramètre effectif et la valeur de **b** est un paramètre formel
☐ aucune de ces 4 réponses.

```
#include <stdio.h>
int a;
void action(int n){
  static int cpt;
  int k;
  n++;  a=12;
  int a=42;
  printf("%d %d %d %d ",n,cpt,k,a);
}
int main(void){
  int b=a++;
  action(b);
  printf("%d %d",a,b);
}
```

10) Qu'affiche-t-il ? ("?" indique une valeur indéfinie - cocher <u>une seule</u> bonne réponse)

☐ `? 0 ? 42 13 ?` ☐ `1 0 ? 42 12 0` ☐ `2 ? 0 42 3 2` ☐ `1 ? ? 12 0 1` ☐ aucune des 4

Exercice 2 (4 points)

On veut écrire un programme de messagerie permettant à des utilisateurs distants de s'envoyer des messages. Pour cela, on définit les structures suivantes :

```
typedef struct date {
       int jour, mois, annee,
           heure, minute, seconde;
} date;
typedef struct utilisateur{
    char nom[50], email[40], mdpasse[20];
    date inscript;
} utilisateur;
```

```
typedef struct message{
    utilisateur *exped, *destin;
    char contenu[500];
    date date_message;
} message;
```

1) Pour écrire un nouveau message, on utilise une fonction qui a le prototype suivant :

```
void saisir_message(message *msg, utilisateur *exp, utilisateur *dest);
```

Dans cette fonction, on trouve un appel à scanf qui permet de saisir l'année du message. Comment peut-il être écrit ? (Cochez <u>2</u> réponses justes, ou une seule si vous n'êtes pas sûr !)

☐ `scanf("%d",&msg.date_message.annee);`
☐ `scanf("%d",&msg->date_message.annee);`
☐ `scanf("%d",*msg.date_message->annee);`
☐ `scanf("%d",&(*msg).date_message.annee);`
☐ `scanf("%d",&(*msg.date_message).annee);`
☐ `scanf("%d",&(*msg.date_message.annee));`
☐ `scanf("%d",&msg->date_message->annee);`

2) Pour afficher un message, on utilise une fonction qui a le prototype suivant :

```
void afficher_message(message msg);
```

Dans cette fonction, on trouve un appel à printf qui permet d'afficher l'année d'inscription de l'expéditeur. Comment peut-il être écrit ? (Cochez <u>2</u> réponses justes, ou une seule si vous n'êtes pas sûr !)

☐ `printf("%d",msg.exped.inscript.annee);`
☐ `printf("%d",msg->exped->inscript.annee);`
☐ `printf("%d",msg.exped->inscript.annee);`
☐ `printf("%d",msg->exped.inscript.annee);`
☐ `printf("%d",(*msg).exped.inscript.annee);`
☐ `printf("%d",(*msg.exped).inscript.annee);`
☐ `printf("%d",*msg.exped.inscript->annee);`

3) Que font les instructions suivantes ?

```
for(int i=0;i<n;++i)
    if(strcmp(listemsg[i].exped->nom,"Toto")==0 && listemsg[i].date_message.annee>=2022)
        afficher_message(listemsg[i])
```

☐ modifient l'année de tous les messages de la liste qui sont envoyés par "Toto".
☐ affichent tous les messages qui ont été envoyés par "Toto" à partir de l'année 2022
☐ affichent tous les messages qui n'ont pas été envoyés par "Toto" à partir de 2022
☐ affichent tous les messages qui ont été envoyés par "Toto" et les messages des autres utilisateurs envoyés à partir de 2022
☐ aucune des 4 réponses

Page annexe 1 : Programme "Apprentissage de l'Alphabet"

```
 1 #include <stdio.h>
 2 #include <.....h>
 3 int main(void){
 4    puts("APPRENONS L'ALPHABET\n\nÉcris les 26 lettres de l'alphabet :");
 5    char juste = ...... ;
 6    do {
 7       char reponse = .......(getchar());
 8       if(reponse=='\n')
 9          if(juste=='A')
10             puts("Vas-y, j'attends que tu tapes les lettres de l'alphabet, de A à Z.");
11          else
12             puts("Bien, continue !");
13       ......... (reponse==juste)
14          ........;
15       else {
16          if(! .........(reponse))
17             printf("\"%c\" n'est pas une lettre de l'alphabet.\n",reponse);
18          if(juste=='A')
19             printf("La première lettre est A");
20          else
21             printf("Après %c, il y a %c", ....... ,juste);
22          printf(", pas \"%c\".\nEssaye encore, depuis le début :\n",.......);
23          while(getchar()!='\n') ;
24          juste='A';
25       }
26    } ......... (juste<='Z');
27    if(getchar()!=......){
28       puts("Z est la dernière lettre. Il n'y a rien après !");
29       while(getchar()!='\n') ;
30    }
31    puts("Très bien, tu connais tout l'alphabet, de A à Z.");
32 }
```

Exemple d'exécution
(lignes soulignées : tapées par l'enfant)

```
APPRENONS L'ALPHABET

Écris les 26 lettres de l'alphabet :
HBCD
La première lettre est A, pas "H".
Essaye encore, depuis le début :
ABCD
Bien, continue !
Efjhi
Après F, il y a G, pas "J".
Essaye encore, depuis le début :
abcdefghijqlnmop
Après J, il y a K, pas "Q".
Essaye encore, depuis le début :
abcdefghijklnmop
Après L, il y a M, pas "N".
Essaye encore, depuis le début :
abcdefghijklmnopqrstuvwxyz
Très bien, tu connais tout l'alphabet, de A à Z.
```

Morceaux à utiliser pour compléter le programme

- a : `'\n'`
- b : `'A'`
- c : `ctype`
- d : `else if`
- e : `isalpha`
- f : `juste++`
- g : `juste-1`
- h : `reponse`
- i : `toupper`
- j : `while`

Page annexe 2 : Fonctions de la bibliothèque standard (Rappels)

```
#include <string.h>

 char *strcat(char *dest, const char *src);
       Ajoute la chaîne src à la chaîne dest et renvoie un pointeur dest.

 char *strchr(const char *s, int c);
       Renvoie un pointeur sur la première occurrence du caractère c dans la chaîne s.

 int strcmp(const char *s1, const char *s2);
       Compare les chaînes s1 et s2.

 char *strcpy(char *dest, const char *src);
       Copie  la  chaîne  src  dans  dest et renvoie un pointeur sur le début de la chaîne
       dest.

 size_t strlen(const char *s);
       Renvoie la longueur de la chaîne s.

 char *strncat(char *dest, const char *src, size_t n);
       Ajoute  au  plus  n  caractères  de  la  chaîne  src à la chaîne dest et renvoie un
       pointeur vers dest.

 int strncmp(const char *s1, const char *s2, size_t n);
       Compare au plus n octets des chaînes s1 et s2.

 char *strncpy(char *dest, const char *src, size_t n);
       Copie au plus n octets de la chaîne src dans dest et renvoie un  pointeur  vers  le
       début de dest.

 char *strrchr(const char *s, int c);
       Renvoie un pointeur sur la dernière occurrence du caractère c dans la chaîne s.

 char *strstr(const char *meule_de_foin, const char *aiguille);
       Recherche la première occurrence de la sous-chaîne aiguille au sein  de  la  chaîne
       meule_de_foin et renvoie un pointeur sur la sous-chaîne trouvée.

#include <ctype.h>

 isalpha(c)
       vérifie si c est un caractère alphabétique. C'est équivalent à (isupper(c) || islower(c)).

 isblank(c)
       vérifie si c est un blanc, c'est-à-dire une espace ou une tabulation.

 isdigit(c)
       vérifie si c est un chiffre (de 0 à 9).

 islower(c)
       vérifie si c est une lettre minuscule.

 isupper(c)
       vérifie si c est une lettre majuscule.


 toupper(c), tolower(c) :

       Ces fonctions convertissent les lettres minuscules en majuscules, et vice versa.

    Si c est une lettre minuscule, toupper() renvoie  son  équivalent  en  majuscule. Sinon, elle
    renvoie c.
    Si  c  est  une  lettre  majuscule,  tolower() renvoie son équivalent en minuscule.  Sinon,  elle
    renvoie  c.
```

Université de Tlemcen
Faculté des Sciences
1re année Licence MI

Corrigé-type de l'Examen Final d'Algorithmique2

Année 2022-2023
Semestre 2
22/05/2023
Durée : 1h30

Exercice 1 (5 pts) Cet exercice compte aussi pour des points bonus au contrôle

Le programme dont le code source est en page annexe1 vous permettra de faire réviser l'alphabet à votre petite sœur ou petit frère. Il lui manque juste les 10 petits morceaux que vous voyez sous le programme. Indiquez ci-dessous **le numéro de ligne** où il faut placer chaque morceau de code pour compléter le programme :

a : 27	**b : 5**	**c : 2**	**d : 13**	**e : 16**	**f : 14**	**g : 21**	**h : 22**	**i : 7**	**j : 26**

Questions diverses (11 pts)

1) Si t est défini par `int t[2][3]={{3,4,5}{0,1,2}} ;` Il manque une virgule au milieu. Si on l'ajoute, on obtient **5**
alors t[0][2] vaut : ☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☒ **5** ☒ **erreur** ⇒ les 2 réponses sont acceptées

2) Qu'affichent les instructions suivantes ?

```
char s1[]="Caractere";
char *s2="Salutations!";
printf("%d %d %d %d", sizeof s1, strlen(s1), strlen(s2), (sizeof s2>sizeof s1));
```

☐ 9 9 12 1 ☐ 9 10 13 1 ☒ **10 9 12 0** ☐ 10 9 12 1 ☐ 9 9 12 0 ☐ aucune des 5

3) Observez les instructions ci-dessous :

```
struct objet{int x,y;};
.....
int b[3]={10,20,30};
struct objet w={10,20};
a=b; v=w;
a[1]=42; v.y=42;
printf("%d %d", b[1], w.y);
```

a) Avec quoi peut-on les compléter (2e ligne) pour qu'elles se compilent sans erreurs ? (cochez une seule réponse)

☐ `int a[3]; struct objet v;`
☐ `int a[]; struct objet *v;`
☒ **`int *a; struct objet v;`**
☐ `int a[3]; struct objet *v;`
☐ `int *a; struct objet *v;`

b) Qu'affichent-elles ? (une seule réponse)

☐ `42 42` ☒ **`42 20`** ☐ `0 20` ☐ `20 42` ☐ `20 20` ☐ `Aucune des` ~~`45`~~ oups !

4) L'appel à la fonction `fopen("mondossier/fichier.txt", "w");` permet d'ouvrir le fichier "fichier.txt" en écriture seule. Dans quels cas peut-il renvoyer la valeur NULL ? (cochez la ou les réponses justes)

☐ Si la taille du fichier est zéro.
☐ Si le fichier n'existe pas dans "mondossier".
☒ **Si "mondossier" n'existe pas.**
☒ **Si le disque qui contient le fichier est endommagé.**
☐ Aucun des 4 cas.

5) Quelles instructions sont équivalentes à l'instruction `puts(chaine);` Cochez la ou les réponses justes :

☐ `fputs(chaine, stdin);`
☒ **`fputs(chaine, stdout);`**
☐ `fputs(chaine, screen);`
☒ **`fprintf(stdout, "%s\n",chaine);`**
☐ `fprintf(stdin, "%s\n",chaine);`
☐ `fscanf(stdout, "%s\n", chaine);`
☐ `fscanf(stdin, "%s\n", chaine);`

6) Soit la déclaration suivante :

```
int a=42, *p=&a, *q=&p[0];
```

Que vaut l'expression `(*q)` ? (cocher une seule bonne réponse)

☐ L'adresse de p ☐ l'adresse de p[0] ☒ **la valeur de a, c'est-à-dire 42**
☐ *p, c'est-à-dire l'adresse de a ☐ aucune des 4

7) Soit la déclaration suivante :

```
int a[]={10,20,30}, *p=&a[0], *q=&p[0];
```

Que vaut l'expression `(*q)` ? (cocher une seule réponse)

☐ L'adresse de p ☐ l'adresse de p[0]
☒ **la valeur de a[0], c'est-à-dire 10**
☐ *p, c'est-à-dire l'adresse de a[0] ☐ aucune des 4

8) Soit la déclaration suivante :

```
int m[2][3]={{10,20,30},{40}};
```

Que vaut l'expression **(m+1) ?(cocher une seule réponse)

☐ 10 ☐ 20 ☒ **40**
☐ 0 ☐ une valeur indéfinie ☐ aucune des 5 réponses

9) Dans le programme ci-contre : (une seule bonne réponse)

☐ la variable **a** est globale, **n** et **cpt** sont statiques, **b** est locale.
☐ **a**, **b**, **n** et **k** sont des variables automatiques et **cpt** est statique
☒ **il y a 2 variables appelées a, l'une est statique, l'autre automatique**
☐ **n** est un paramètre effectif et la valeur de **b** est un paramètre formel
☐ aucune de ces 4 réponses.

```
#include <stdio.h>
int a;
void action(int n){
  static int cpt;
  int k;
  n++;  a=12;
  int a=42;
  printf("%d %d %d %d ",n,cpt,k,a);
}
int main(void){
  int b=a++;
  action(b);
  printf("%d %d",a,b);
}
```

10) Qu'affiche-t-il ? ("?" indique une valeur indéfinie - cocher une seule bonne réponse)

☐ ? 0 ? 42 13 ? ☒ **1 0 ? 42 12 0** ☐ 2 ? 0 42 3 2 ☐ 1 ? ? 12 0 1 ☐ aucune des 4

Exercice 2 (4 points)

On veut écrire un programme de messagerie permettant à des utilisateurs distants de s'envoyer des messages. Pour cela, on définit les structures suivantes :

```
typedef struct date {
       int jour, mois, annee,
           heure, minute, seconde;
} date;
typedef struct utilisateur{
    char nom[50], email[40], mdpasse[20];
    date inscript;
} utilisateur;
```

```
typedef struct message{
    utilisateur *exped, *destin;
    char contenu[500];
    date date_message;
} message;
```

1) Pour écrire un nouveau message, on utilise une fonction qui a le prototype suivant :

```
void saisir_message(message *msg, utilisateur *exp, utilisateur *dest);
```

Dans cette fonction, on trouve un appel à scanf qui permet de saisir l'année du message. Comment peut-il être écrit ? (Cochez 2 réponses justes, ou une seule si vous n'êtes pas sûr !)

☐ `scanf("%d",&msg.date_message.annee);`
☒ **`scanf("%d",&msg->date_message.annee);`**
☐ `scanf("%d",*msg.date_message->annee);`
☒ **`scanf("%d",&(*msg).date_message.annee);`**
☐ `scanf("%d",&(*msg.date_message).annee);`
☐ `scanf("%d",&(*msg.date_message.annee));`
☐ `scanf("%d",&msg->date_message->annee);`

2) Pour afficher un message, on utilise une fonction qui a le prototype suivant :

```
void afficher_message(message msg);
```

Dans cette fonction, on trouve un appel à printf qui permet d'afficher l'année d'inscription de l'expéditeur. Comment peut-il être écrit ? (Cochez 2 réponses justes, ou une seule si vous n'êtes pas sûr !)

☐ `printf("%d",msg.exped.inscript.annee);`
☐ `printf("%d",msg->exped->inscript.annee);`
☒ **`printf("%d",msg.exped->inscript.annee);`**
☐ `printf("%d",msg->exped.inscript.annee);`
☐ `printf("%d",(*msg).exped.inscript.annee);`
☒ **`printf("%d",(*msg.exped).inscript.annee);`**
☐ `printf("%d",*msg.exped.inscript->annee);`

3) Que font les instructions suivantes ?

```
for(int i=0;i<n;++i)
    if(strcmp(listemsg[i].exped->nom,"Toto")==0 && listemsg[i].date_message.annee>=2022)
        afficher_message(listemsg[i])
```

☐ modifient l'année de tous les messages de la liste qui sont envoyés par "Toto".
☒ **affichent tous les messages qui ont été envoyés par "Toto" à partir de l'année 2022**
☐ affichent tous les messages qui n'ont pas été envoyés par "Toto" à partir de 2022
☐ affichent tous les messages qui ont été envoyés par "Toto" et les messages des autres utilisateurs envoyés à partir de 2022
☐ aucune des 4 réponses

Page annexe 1 : Programme "Apprentissage de l'Alphabet"

```
 1 #include <stdio.h>
 2 #include <.....h>
 3 int main(void){
 4    puts("APPRENONS L'ALPHABET\n\nÉcris les 26 lettres de l'alphabet :");
 5    char juste = ...... ;
 6    do {
 7       char reponse = .......(getchar());
 8       if(reponse=='\n')
 9          if(juste=='A')
10             puts("Vas-y, j'attends que tu tapes les lettres de l'alphabet, de A à Z.");
11          else
12             puts("Bien, continue !");
13       ......... (reponse==juste)
14          ........;
15       else {
16          if(! .........(reponse))
17             printf("\"%c\" n'est pas une lettre de l'alphabet.\n",reponse);
18          if(juste=='A')
19             printf("La première lettre est A");
20          else
21             printf("Après %c, il y a %c", ....... ,juste);
22          printf(", pas \"%c\".\nEssaye encore, depuis le début :\n",.......);
23          while(getchar()!='\n') ;
24          juste='A';
25       }
26    } ......... (juste<='Z');
27    if(getchar()!=......){
28       puts("Z est la dernière lettre. Il n'y a rien après !");
29       while(getchar()!='\n') ;
30    }
31    puts("Très bien, tu connais tout l'alphabet, de A à Z.");
32 }
```

Exemple d'exécution
(lignes soulignées : tapées par l'enfant)

```
APPRENONS L'ALPHABET

Écris les 26 lettres de l'alphabet :
HBCD
La première lettre est A, pas "H".
Essaye encore, depuis le début :
ABCD
Bien, continue !
Efjhi
Après F, il y a G, pas "J".
Essaye encore, depuis le début :
abcdefghijqlnmop
Après J, il y a K, pas "Q".
Essaye encore, depuis le début :
abcdefghijklnmop
Après L, il y a M, pas "N".
Essaye encore, depuis le début :
abcdefghijklmnopqrstuvwxyz
Très bien, tu connais tout l'alphabet, de A à Z.
```

Morceaux à utiliser pour compléter le programme

- a : `'\n'`
- b : `'A'`
- c : `ctype`
- d : `else if`
- e : `isalpha`
- f : `juste++`
- g : `juste-1`
- h : `reponse`
- i : `toupper`
- j : `while`

Page annexe 2 : Fonctions de la bibliothèque standard (Rappels)

```
#include <string.h>

 char *strcat(char *dest, const char *src);
       Ajoute la chaîne src à la chaîne dest et renvoie un pointeur dest.

 char *strchr(const char *s, int c);
       Renvoie un pointeur sur la première occurrence du caractère c dans la chaîne s.

 int strcmp(const char *s1, const char *s2);
       Compare les chaînes s1 et s2.

 char *strcpy(char *dest, const char *src);
       Copie  la  chaîne  src  dans  dest et renvoie un pointeur sur le début de la chaîne
       dest.

 size_t strlen(const char *s);
       Renvoie la longueur de la chaîne s.

 char *strncat(char *dest, const char *src, size_t n);
       Ajoute  au  plus  n  caractères  de  la  chaîne  src à la chaîne dest et renvoie un
       pointeur vers dest.

 int strncmp(const char *s1, const char *s2, size_t n);
       Compare au plus n octets des chaînes s1 et s2.

 char *strncpy(char *dest, const char *src, size_t n);
       Copie au plus n octets de la chaîne src dans dest et renvoie un  pointeur  vers  le
       début de dest.

 char *strrchr(const char *s, int c);
       Renvoie un pointeur sur la dernière occurrence du caractère c dans la chaîne s.

 char *strstr(const char *meule_de_foin, const char *aiguille);
       Recherche la première occurrence de la sous-chaîne aiguille au sein  de  la  chaîne
       meule_de_foin et renvoie un pointeur sur la sous-chaîne trouvée.

#include <ctype.h>

 isalpha(c)
       vérifie si c est un caractère alphabétique. C'est équivalent à (isupper(c) || islower(c)).

 isblank(c)
       vérifie si c est un blanc, c'est-à-dire une espace ou une tabulation.

 isdigit(c)
       vérifie si c est un chiffre (de 0 à 9).

 islower(c)
       vérifie si c est une lettre minuscule.

 isupper(c)
       vérifie si c est une lettre majuscule.


 toupper(c), tolower(c) :

       Ces fonctions convertissent les lettres minuscules en majuscules, et vice versa.

    Si c est une lettre minuscule, toupper() renvoie  son  équivalent  en  majuscule. Sinon, elle
    renvoie c.
    Si  c  est  une  lettre  majuscule,  tolower() renvoie son équivalent en minuscule.  Sinon,  elle
    renvoie  c.
```