INF203 - Partiel
Année 2022-2023

Durée : 2h.
Une feuille A4 manuscrite recto/verso autorisée, dictionnaire papier (non annoté) autorisé pour les étudiants étrangers uniquement.
Tout autre document, calculatrices et appareils électroniques interdits.
Pour chaque question, une partie des points peut tenir compte de la présentation.
Le barème est indicatif.

Toute réponse même incomplète sera valorisée à partir du moment où elle réalise au moins une partie de ce qui est demandé. Les questions sont relativement indépendantes, et dans tous les cas vous pouvez utiliser les scripts et les fonctions demandées dans les questions précédentes même si vous n'avez pas réussi à les écrire.

Par ailleurs, tout code ou algorithme *élégant*, en plus d'être correct, sera bonifié.

1 Automates (4 points)

Romain souhaite installer un digicode à sa porte. Il souhaite mettre le code "2023". Il a le choix entre deux modèles de digicode :

(a) On peut taper des chiffres en continu et la porte s'ouvre dès que les 4 derniers chiffres tapés sont "2023", autrement rien ne se passe.

Attention : Bien penser à ce qui se produit pour ce modèle lorsque l'utilisateur tape 2-0-2-0-2-3 par exemple.

(b) L'appareil attend une séquence de 4 chiffres et réagit après coup. Si les 4 chiffres sont "2023", la porte s'ouvre. Sinon, l'appareil produit un bruit d'échec "bip-bip" et attend à nouveau une séquence de 4 chiffres.

Les deux modèles d'appareil font "biiip" quand la porte s'ouvre après un code correct.

Question :

1. (4 points) Dessiner les automates des deux modèles de digicodes proposés.

On admettra les conventions suivantes :

— Le clavier comporte uniquement les 10 touches correspondant aux 10 chiffres $0, 1, 2, \ldots$

— Au début du fonctionnement, aucun chiffre n'a encore été tapé.

— On considère l'automate dans un état final une fois la porte ouverte.

— On notera $[\neq 1]$ pour désigner l'entrée "n'importe quel chiffre sauf le chiffre 1" (et pareillement pour les autres chiffres).

— On notera aussi $[\neq 1, 2]$ pour "n'importe quel chiffre sauf 1 et 2", $[\neq 1, 2, 3]$ pour "n'importe quel chiffre sauf 1, 2 et 3", etc...

— * désignera l'entrée "n'importe quel chiffre".

— Si rien ne se passe, on ne précisera pas de sortie (pour ne pas surcharger le dessin).

2 Programmation en langage C : Shifumi (8 points)

On souhaite écrire un programme pour le jeu Shifumi.

Les règles du jeu sont les suivantes : deux joueurs choisissent chacun en secret une action parmi PAPIER, CAILLOU ou CISEAUX et révèlent leur choix simultanément. Si leurs choix sont identiques, il y a égalité. Sinon :

— le PAPIER bat le CAILLOU ;

— le CAILLOU bat les CISEAUX ;

— les CISEAUX battent le PAPIER.

Le jeu se joue en plusieurs manches successives si nécessaire : le gagnant est le premier joueur à remporter une manche (tant que les joueurs font la même action, on joue une manche supplémentaire).

Pour cela, on définit tout d'abord un type `action` (synonyme du type `int`), on considère les actions décrites comme ci-dessous et on fixe un nombre de manches maximal que durera la partie :

```
typedef action int;
#define PAPIER 0
#define CAILLOU 1
#define CISEAUX 2
#define NBMANCHES 10
```

Questions :

1. (1 point) Définir la structure *joueur* composée des champs suivants :
   — *nom* : chaîne de caractères qui donne le nom du joueur ;
   — *actions* : un tableau de NBMANCHES entiers qui représente la suite des actions (parmi PAPIER, CAILLOU, CISEAUX) jouées par le joueur au cours d'une partie ;
2. (1 point) On suppose qu'on dispose d'une fonction

   ```
   int generer_entier(int borne);
   ```

   qui génère un entier aléatoire entre 0 et borne-1.
   Écrire la fonction ci-dessous, qui tire au hasard la liste des coups que va jouer le joueur `j` pour sa prochaine partie.

   ```
   void choisir_actions(joueur *j);
   ```

3. (2 points) Écrire les deux fonctions ci-dessous, qui décident le résultat d'une manche, pour la première fonction en ne prenant que les actions en arguments, puis pour la seconde en prenant les joueurs et le numéro de la manche.

   ```
   /* resultat_shifumi :
     renvoie 1 si l'action a1 est gagnante sur a2;
     renvoie -1 si l'action a1 est perdante sur a2;
     renvoie 0 sinon.
   */
   int resultat_shifumi(action a1, action a2);

   /* jouer_manche :
     renvoie le résultat de la manche donnée entre les joueurs j1 et j2.
   */
   int jouer_manche(joueur* j1, joueur* j2, int manche);
   ```

4. (2 points) Écrire la fonction

   ```
   joueur* duel(joueur* j1, joueur* j2);
   ```

   qui renvoie un pointeur vers le joueur qui remporte la partie (le premier à faire une action gagnante). En cas d'égalité au bout des NBMANCHES, on renverra le pointeur NULL.
5. (2 points) À partir des fonctions précédentes, on veut dérouler un défi au cours duquel un joueur particulier (le champion) rencontre successivement une série d'adversaires. Pour cela, écrire la fonction ci-dessous :

   ```
   int defi(joueur *champion, joueur* participants[], int m);
   ```

   L'argument `m` est le nombre de participants au défi. Le joueur `champion` rencontre chaque participant, en regénérant une nouvelle liste de coups au hasard entre chaque partie.
   Votre fonction doit renvoyer le nombre de parties remportées par le `champion`.

3 Programmation en langage C et Shell : le Pendu (10 points)

On se propose d'écrire un programme C qui permet de jouer au jeu du Pendu, dans lequel un joueur essaye de deviner un mot en proposant des lettres afin de savoir si celles-ci sont présentes dans le mot à deviner.

Le programme prendra en argument un fichier texte (`.txt`) contenant un unique mot inconnu de l'utilisateur. Le programme demandera à l'utilisateur de taper des lettres au clavier. Les lettres du mot inconnu sont initialement cachées, puis révélées quand l'utilisateur les tape.

Par exemple, si le mot inconnu est "pomme", le programme affichera "?????" puis demandera une lettre à l'utilisateur. Si l'utilisateur tape "m", le programme affichera ensuite "??mm?", et ainsi de suite. Quand l'utilisateur tape une lettre qui n'est pas dans le mot, c'est une erreur. La partie se termine si le mot est entierement révelé (victoire) ou si le nombre d'erreurs atteint 11 (défaite).

3.1 Fonctionnement du jeu

On propose de stocker les données d'une partie en cours dans la structure suivante.

```
typedef struct partie {
    char mot_entier[TAILLE_MAX];
    char mot_masque[TAILLE_MAX];
    int nb_erreurs;
} partie;
```

Le champ `mot_entier` est la chaîne de caractères du mot à trouver (par exemple "pomme"). Le champ `mot_masque` est la chaîne de caractères où seules les lettres tapées par l'utilisateur sont révelées (par exemple "??mm?"). Le champ `nb_erreurs` compte le nombre de lettres déjà tapées par l'utilisateur et qui n'apparaissent pas dans le mot à deviner.

1. (1 point) Écrire la fonction

   ```
   void afficher(partie *p);
   ```

   qui montre l'état actuel de la partie sur la sortie standard.

   Vous êtes libres de choisir l'affichage qui vous semble pertinent. *Attention : ne pas afficher le mot entier !*

2. (2 points) Écrire une fonction

   ```
   void verifier_lettre(partie *p, char lettre);
   ```

   qui doit

   — Modifier le champ `mot_masque` de `p` pour révéler toutes les occurrences de `lettre` dans le mot.

   — Augmenter le nombre d'erreurs de 1 si `lettre` n'apparaît pas dans le mot.

   Contrairement au jeu réel, on ne mémorisera pas les lettres déjà proposées par le joueur. Reproposer une lettre déjà trouvée ne change donc rien à l'état de la partie, mais reproposer une lettre absente du mot augmentera le nombre d'erreurs.

3. (1 point) Écrire une fonction

   ```
   int verifier_fin_partie(partie *p);
   ```

   qui vérifie si la partie est terminée. La valeur renvoyée sera

   — 0 si la partie n'est pas terminée.

   — 1 si la partie se termine par une victoire (mot totalement révélé).

   — 2 si la partie se termine par une défaite (nombre d'erreurs = 11).

   On suppose que la partie reçue en argument ne peut pas comporter à la fois un mot totalement révélé et un nombre d'erreurs supérieur à 10.

3.2 Lecture du mot dans un fichier texte

Le mot à trouver est écrit dans un fichier passé en argument du programme.

4. (2 points) Écrire une fonction

```
int initialiser_partie(partie *p, char *nom_fichier);
```

qui :

- ouvre le fichier dont le nom est donné en argument et lit dans ce fichier le mot à deviner ;
- initialise tous les champs de la partie `p` en se basant sur le mot lu dans le fichier.
- Si le fichier n'existe pas ou n'est pas accessible en lecture, on affichera un message d'erreur et la fonction renverra 1, sinon elle renverra 0.

On rappelle le prototype de la fonction `fscanf` :

```
int fscanf(FILE *flux, char *format, char *destination);
```

3.3 Programme principal

On suppose disposer d'une fonction

```
char demander_lettre();
```

qui demande à l'utilisateur d'entrer une lettre minuscule au clavier et la renvoie (il n'est pas demandé d'implémenter cette fonction).

5. (2 points) Écrire la fonction `main` du programme de jeu du Pendu.
   Votre programme recevra un argument en ligne de commande, qui est le nom du fichier dans lequel on lit le mot à deviner.
   Il permettra ensuite de jouer une partie complète jusqu'à son terme.

   On vérifiera notamment le nombre d'arguments passés au programme, et on gèrera les différents cas d'erreurs possibles.
   Vous êtes libres dans l'affichage des messages dédiés à l'utilisateur au début, pendant et à la fin de la partie.

3.4 Plusieurs parties à la suite

Dans un répertoire `Mots` que vous a envoyé un ami, il y a des fichiers texte de noms `motmystere1.txt`, `motmystere2.txt`, ..., et ainsi de suite jusqu'à `motmystere42.txt`. Votre ami souhaite que vous jouiez au Pendu avec ces mots dans l'ordre. Votre programme compilé, que vous avez appelé `pendu`, et le répertoire `Mots` se trouvent tous deux dans votre répertoire courant.

6. (2 points) Écrivez un script Shell qui lance à la suite les 42 parties de Pendu pour chacun des mots du répertoire `Mots`.

4 INF203 et enjeux sociétaux (Bonus, 2 points)

Question (réponse en 10-20 lignes maximum) :

1. (2 points) Nous vivons à l'aube d'une ère de changements (climatiques, sociétaux, techniques) qu'il s'agira d'affronter avec des compétences d'adaptation nouvelles. Quelles sont les compétences que tu estimes avoir acquises au cours de l'enseignement d'INF203 et qui te donnent l'impression d'être plus "armé·e" face aux enjeux à venir ?