+1/1/60+

UPS – L1 Informatique / MIASHS **30/04/2019 – Session 1**

Programmation en C (EPINF2E1)

Nom et prénom :

.....

- *Durée : 2 heures.*
- *Aucun document n'est autorisé. L'usage de la calculatrice ou de tout équipement électronique est interdit.*
- *Le sujet qui vous a été attribué est unique et il ne vous sera fourni que cet exemplaire.*
- *Cochez ou noircissez l'intérieur des cases choisies. Vous pouvez utiliser un correcteur liquide pour faire disparaître une case noircie par erreur (**mais ne redessinez pas la bordure de la case**).*
- *Les questions faisant apparaître le symbole ♣ peuvent présenter zéro, une ou plusieurs bonnes réponses. Les autres ont une unique bonne réponse. Il n'y a pas de points négatifs entre les questions.*

Codez votre numéro d'étudiant ci-dessous :

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

1 Questionnaire à choix multiples (12 points)

2 Questions ouvertes (8 points) – exemple de corrigé

Nous souhaitons écrire des fonctions qui manipulent un jeu de cartes, ainsi qu'un programme qui implémente un jeu très simple pour tester les fonctions. Nous considérons un jeu de 52 cartes organisées en quatre enseignes (pique, cœur, trèfle, carreau), chacune ayant 13 cartes (de l'as au roi). On utilisera pour cela les définitions suivantes :

```
#define MAXCARTES 52
struct carte {
    char enseigne; /* 'P'=Pique, 'C'=Coeur, 'T'=Trèfle, 'K'=Carreau */
    int valeur; /* 1 à 13 avec 1=As, 11=Valet, 12=Dame, 13=Roi */
};
struct paquet_cartes {
    int nbre; /* nombre de cartes du paquet */
    struct carte cartes[MAXCARTES]; /* les cartes du paquet */
};
```

On suppose par ailleurs les deux fonctions suivantes déjà écrites :

- `void initialiser_jeu(struct paquet_cartes *jc)` qui initialise la structure pointée par `jc` avec les 52 cartes du paquet ;
- `void melanger_jeu(struct paquet_cartes *jc)` qui mélange de manière aléatoire les cartes du paquet pointé par `jc`.

Enfin, pour la question 28, nous supposons que la fonction principale démarre par les déclarations suivantes :

```
int main(void)
{
    struct paquet_cartes jeu; /* jeu de cartes complet */
    struct paquet_cartes j1, j2; /* paquets des joueurs 1 et 2 */
    int pj1 = 0, pj2 = 0; /* points des joueurs 1 et 2 */
    ...
}
```

Q.1) ♣ Écrivez la fonction d'en-tête : `void ajouter_carte(struct paquet_cartes *pc, struct carte c)` qui ajoute la carte `c` à la fin du paquet de cartes pointé par `pc`. On supposera que le nombre de cartes du paquet est strictement inférieur au maximum avant l'appel de cette fonction. Écrivez ensuite la fonction d'en-tête : `struct carte retirer_carte(struct paquet_cartes *pc)` qui retire et renvoie la dernière carte du paquet. On supposera qu'il y a au moins une carte dans le paquet avant l'appel de cette fonction.

```
/* pre-condition : pc->nbre < MAXCARTES */
void ajouter_carte(struct paquet_cartes *pc, struct carte c) {

    pc->cartes[pc->nbre] = c;
    pc->nbre++;
}

/* pre-condition : pc->nbre > 0 */
struct carte retirer_carte(struct paquet_cartes *pc) {

    struct carte c = pc->cartes[pc->nbre-1];
    pc->nbre--;
    return c;
}
```

Q.2) ♣ Écrivez la fonction d'en-tête `int meilleure_carte(struct carte c1, struct carte c2)` qui renvoie 0 si les cartes `c1` et `c2` ont la même valeur, 1 si `c1` est plus grande que `c2` et -1 si `c2` est plus grande que `c1`. Notez que dans le jeu l'as (valeur 1) est la plus grande carte, suivie du roi, de la dame, ... jusqu'au 2.

```
int meilleure_carte(struct carte c1, struct carte c2) {

    if (c1.valeur == c2.valeur)
        return 0;
    if (c1.valeur == 1 || (c1.valeur > c2.valeur && c2.valeur != 1))
        return 1;
    else
        return -1;
}
```

Q.3) ♣ Écrivez la fonction d'en-tête `void distribuer_2joueurs(struct paquet_cartes *jeu, struct paquet_cartes *j1, struct paquet_cartes *j2)` qui distribue toutes les cartes du paquet pointé par `jeu` dans les paquets pointés par `j1` et `j2`. La distribution se fera carte par carte, en alternant une carte pour chaque joueur. Pour écrire cette fonction, vous ferez appel aux deux fonctions de la question 25.

```
void distribuer_2joueurs( struct paquet_cartes *jeu,
                          struct paquet_cartes *j1,
                          struct paquet_cartes *j2 ) {

    while (jeu->nbre > 0) {
        ajouter_carte(j1, retirer_carte(jeu));
        if (jeu->nbre > 0)
            ajouter_carte(j2, retirer_carte(jeu));
    }
}
```

Q.4) ♣ En utilisant toutes les fonctions précédentes, complétez la fonction principale (dont le début se trouve dans l'énoncé) de telle sorte à : initialiser et mélanger le jeu de cartes `jeu` ; distribuer le jeu aux deux joueurs (`j1` et `j2`) ; et comparer chaque carte des joueurs, une par une, en accordant un point au joueur qui a la plus grande carte. Lorsqu'il n'y a plus de carte, le joueur qui a le plus de points est affiché, ou la chaîne de caractères « égalité ».

```
int main(void) {

    struct paquet_cartes jeu; /* jeu de cartes complet */
    struct paquet_cartes j1, j2; /* mains des joueurs 1 et 2 */
    int pj1 = 0, pj2 = 0; /* points des joueurs 1 et 2 */

    struct carte cj1, cj2;
    int res;

    initialiser_jeu(&jeu);
    melanger_jeu(&jeu);
    distribuer_2joueurs(&jeu, &j1, &j2);

    /* note : une condition suffirait ici car j1.nbre == j2.nbre */
    while (j1.nbre>0 && j2.nbre>0) {
        cj1 = retirer_carte(&j1);
        cj2 = retirer_carte(&j2);
        res = meilleure_carte(cj1, cj2);
        if (res==1)
            pj1++;
        else if (res==-1)
            pj2++;
    }

    if (pj1>pj2)
        printf("Le joueur 1 gagne ! \n");
    else if (pj2>pj1)
        printf("Le joueur 2 gagne ! \n");
    else
        printf("Egalite ! \n");

    return 0;
}
```