efrei PARIS SUD

Techniques de Progra
L1 –PL1 – CPI1

16,5

BLOQUET
Romain

PL1
2013

Devoir Ecrit

Le Vendredi 23 mai 2014

Durée : 2h

Sans documents, sans calculatrice

Avant de débuter le DE :

- Prenez le temps de bien lire les énoncés des exercices. Vous répondrez dans les espaces laissés libres à cet effet dans le sujet. Aucune autre copie ne sera prise en compte.
- N'oubliez pas de reporter vos nom, prénom, groupe et école dans les espaces ci-dessous
- **La qualité de la rédaction de vos réponses entrera en compte pour la notation des exercices**

NOM Bloquet

PRENOM Romain

GROUPE PL1

Promotion L1/PL1 [X] CPI1 [ ]

Thème 1 : Structures

a) En langage C, que signifie **typedef** ?

typedef sert a créer un raccourcit du nom de la structure crée. Car si on marque struct S_nom_structure on sera obligé de marquer ça a chaque fois pour déclarer une variable de type de la structure. Alors qu'avec typedef on marque t_nom_structure.

(1)

b) Définissez une structure permettant de définir une personne dans le cadre d'une application bancaire.

```
Structure t_personne
{
caractère * nom;
caractère * prenom;
entier numero_compte;
réel montant_du_compte;
réel operation_a_venir;
};
```

(1)

c) Soit le programme suivant, à gauche duquel on a porté les numéros des lignes :

```
1     structure t_complexe
2     {
3           reel re,im;
4     };
5     fonction principale()
6     {
7           t_complexe z1, z2;
8
9           z1 ← 1+3i; X
10          z2.re ← 5.2;
11          t_complexe.im ← 4; X
12          z1.re ← z2.re;
13          z1.im ← 1;
14
15          z2 ← z1+z2; X
16
17          afficher(z2); X
18          retourner;
19    }
```

Indiquez les lignes incorrectes de ce programme en expliquant pourquoi elles sont incorrectes (réponse page suivante)

ligne 9 : On ne peut pas directement remplir le z1 complexe de cette maniere car il faut remplir chaque champs un par un. par exemple z1.re ← 1 et z1.im ← 3.
On peut créer une fonction pour faire cela.

ligne 11 : la notation point est utilisé pour remplir le champ d'une variable : nom_de_la_variable . nom_du_champ or ici il est question de remplir le champ im du type t_complexe, ce qui ne signifie rien.

ligne 15 : On ne peut pas additionner de Variable de type t_complexe de cette maniere, il faut additioner le champ re de z1 et z2 et le champ im de z1 et z2. Pour ce faire il faut donc créer une fonction.

ligne 17 : pour afficher cette variable il faudra créer une fonction qui affichera le champ re de z2, "+", le champs im de z2, "i".

(2)

d) Soit la structure suivante décrivant une heure

```
structure t_heure
{
      entier hh, mm, ss;
};
```

e) Ecrivez les fonctions suivantes :

- Affichage d'une heure
- Saisie d'une heure
- Comparaison de deux heures
- Addition de deux heures

Affichage :

entrée :

```
fonction afficheHeure (t_heure heure)
{
    afficher ("il est : ", heure.hh, "h", heure.mm,
              "min", heure.ss, "s");

    retour;
}
```

(1)

Saisie :

entrée :

```
fonction SaisieHeure (t_heure *heure)
{
    afficher ("Veuillez saisir l'heure : ");
    saisir (heure -> hh);
    afficher ("Veuillez saisir les minutes : ");
    saisir (heure -> mm);
    afficher ("Veuillez saisir les secondes : ");
    saisir (heure -> ss);
    retour;
}
```

(1)

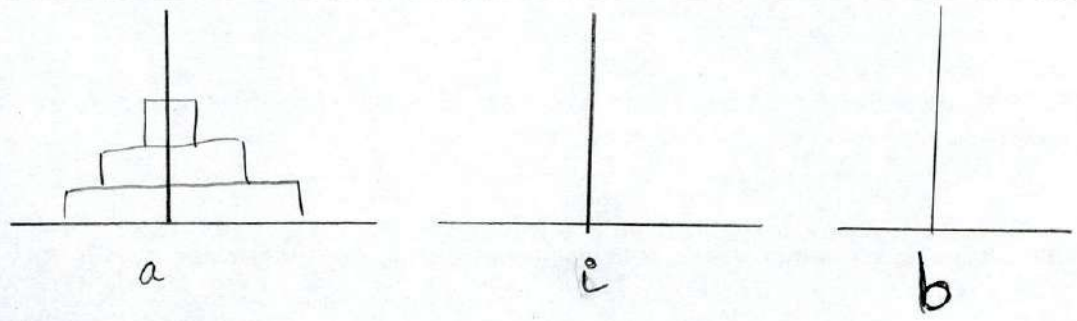
Comparaison :

entrées :

```
fonction Comparaison (t_heure heure_1, t_heure heure_2)
{
    Si (heure_1.hh = heure_2.hh et heure_1.mm = heure_2.mm
        et heure_1.ss = heure_2.ss) alors
    {
        Afficher ("les heures sont identiques");
    }
    Sinon
    {
        Afficher ("les heures sont differentes");
    }
    retour;
}
```

(1)

k) Ecrire une fonction récursive qui résout le problème des tours de Hanoi pour N disques (vu en cours) – vous pouvez faire des schémas si vous le souhaitez pour vous aider à écrire cette fonction



fonction hanoi (entree: Entier nb_disque, entier a, entier i, entier b)

{

0,5

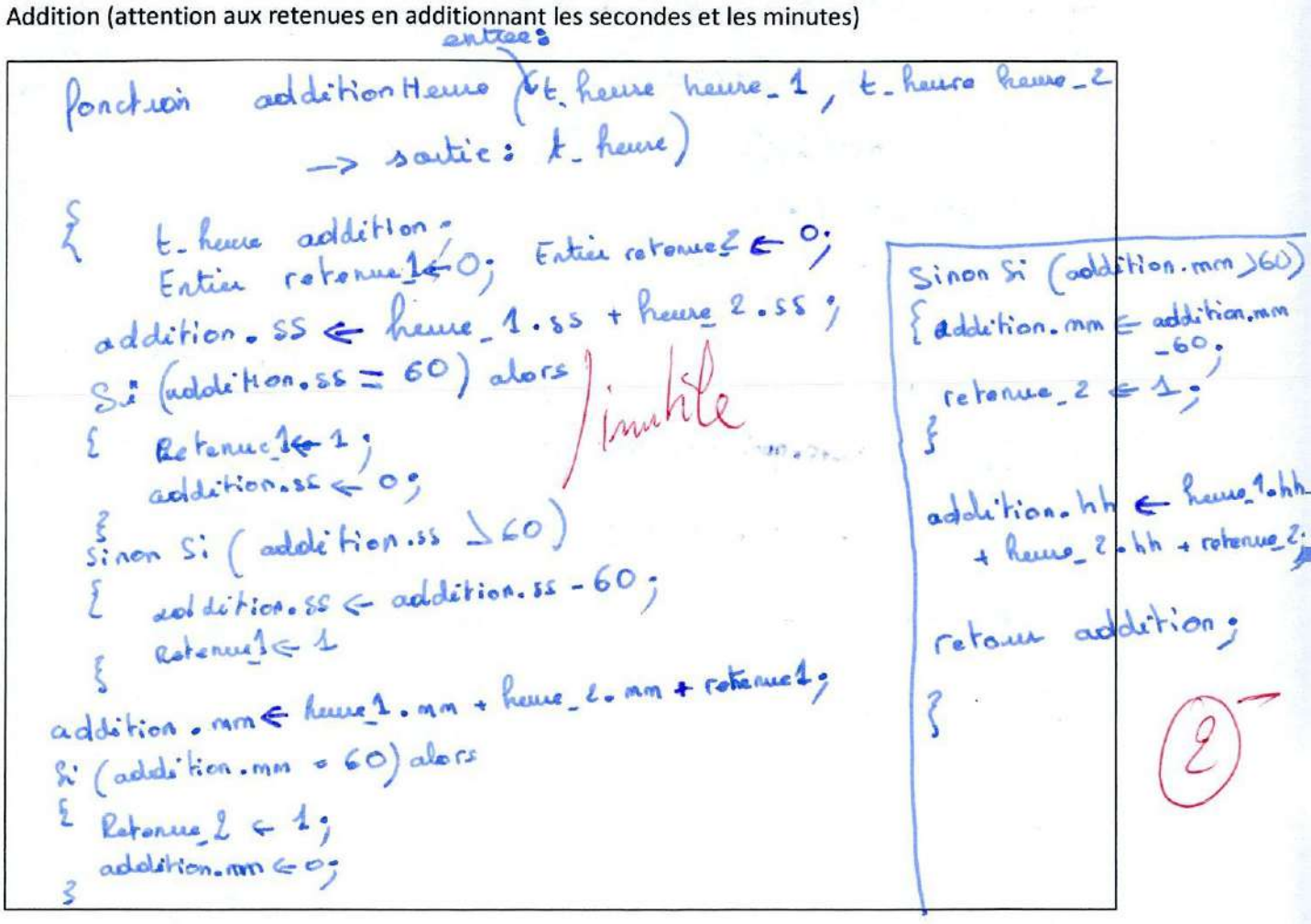
Thème 3 : VRAI/FAUX

j) VRAI/FAUX : répondez par *vrai* ou par *faux* aux affirmations suivantes. Une bonne réponse rapporte 1 point.

Une fonction récursive sans condition d'arrêt peut générer un plantage du programme.	Vrai
Une structure contient des champs.	Vrai
Un champ est identifié par son type et son nom.	Vrai
On utilise la notation '.' pour accéder au champ d'une structure si ce champ est un pointeur.	~~Faux~~
Une variable de type structure n'a pas d'adresse.	faux

2

Addition (attention aux retenues en additionnant les secondes et les minutes)



f) Soit la structure suivante décrivant un événement :

```
structure t_evt
{
        entier   jour,mois,annee;
        caractere       *quoi;
        entier   importance;
};
```

Ecrivez la fonction de saisie d'un événement pour laquelle on vous fournit l'entête suivante :

```
fonction saisieEvt(entree : t_evt *evt)
```

note : cette fonction est appelée dans la fonction principale de la manière suivante :

```
fonction principale()
{
        t_evt e;
        saisieEvt(&e);
        ...
}
```

fonction

```
SaisieEvt (entree: t_evt * evt)
{
  chaine tampon [1000];
  saisir ((*evt).jour);   // peut s'ecrire evt -> jour.
  saisir ((*evt).mois);
  saisir ((*evt).annee);
  saisir (tampon);
  evt.quoi ← reservation ((longueur texte (tampon)+1) caractère);
  copier ((*evt.quoi), tampon);
  saisir ((*evt).importance);
  retour; evt -> importance
}
```

g) Soit **tabag** un tableau (statique ou dynamique, peu importe) de **t_evt** dont la taille utile est connue. Ecrire une fonction qui trie un tableau de **t_evt** par ordre chronologique. L'entête de cette fonction est la suivante :

```
fonction triChrono(entree : t_evt *tab, entier util)
```

Note : cette fonction est appelée dans la fonction principale de la manière suivante :

```
fonction principale()
{
    t_evt     *tabag;
    entier    nbEvt;
    entier    cpt;

    afficher("entrez le nombre d'evenements:");
    saisir(nbEvt);

    tabag ← reservation(nbEvt t_evt);
```

```
        pour cpt de 0 à nbEvt-1
        {
              saisieEvt(&tabag[cpt]);
        }

        triChrono(tabaf,nbEvt);
        ...
}
```

```
fonction triChrono (entree : t_evt * tab, entier util)
{
  entier cpt; tev_t *temp;
  Pour (cpt de 0 à util-2)  ) boucles
  {
    Si (tab[cpt].annee > tab[cpt+1].annee) alors
    {
        temp ← tab[cpt]      ;
        tab[cpt]     ← tab[cpt+1]    ;
        tab[cpt+1]   ← temp;

    }
  }
  retour;
}
```

18

Thème 2 : un peu de récursivité

h) Qu'est-ce qu'une fonction récursive ?

une fonction récursive est une fonction qui fait appel a elle même dans son corps de fonction.
par exemple :

```
fonction facto (entree : entier n -> sortie entier)
{
  si (n = 1)
  { n <- 1 ;
  }
  si (n > 1)
  { n <- n * facto (n-1);
  }
  retour n ;
}
```

(1)

i) Soit la fonction récursive suivante : que fait-elle si on l'appelle avec une valeur initiale de position égale à 0 ? (vous pouvez choisir un texte à afficher en exemple)

```
fonction devinette(entree : caractere *texte, entier position)
{
      si ( position < longueur_texte(texte))
      {
            afficher(texte[position]);
            devinette(texte, position+1);
      }

      retourner;
}
```

Mot choisi : programme. (3 charactères)
la fonction va démmarer.
en 1er : elle affiche "p"
en 2ème elle est rappellée et la position est égale a 1
donc elle affiche "r"
le 1er appel est en pause le temps que le resultat soit obtenue.
3ème : une fois le deuxième appel terminer le mot programme est afficher et la fonction est arrêté grace a la condition (position (2) longueur_texte(texte))

(1)