Nom :	Prénom :
N° Étudiant :	Groupe de TD :

TME Solo 2020 – 2021 – Sujet n°2

Architecture des ordinateurs 1 – LU3IN029

Durée : 0h50

Documents autorisés : Aucun document ni machine électronique n'est autorisé à l'exception du mémento MIPS.

Le barème indiqué pour chaque question n'est donné qu'à titre indicatif. Le barème total est lui aussi donné à titre indicatif. Merci de rendre la feuille.

Étapes préliminaires à suivre scrupuleusement :

1. Créez un répertoire pour le TME solo à la racine de votre compte qui devra contenir les codes réalisés, en tapant les commandes suivantes dans un terminal :
   `> cd`
   `> mkdir TMESolo_<nom>` (où <nom> est votre nom en minuscule sans espace ni accent)
   `> chmod -R go-rwx .`
   Attention : cette dernière commande est très importante car elle empêche d'autres utilisateurs d'accéder à vos fichiers. Si vous ne la faites pas correctement et qu'un autre étudiant copie vos fichiers, vous risquez d'obtenir la note de 0.
   Remarque : la détection de plagiat sera faite automatiquement par logiciel.
2. Ouvrez un éditeur de texte et sauvegardez le fichier `~/TMESolo_<nom>/rapport.txt`. Ce fichier contiendra votre rapport de TME solo. Vous devrez y copier-coller tous vos codes réponses. Vous indiquerez bien le numéro de la question auquel se rapporte chaque code.
   Attention : ce fichier doit se trouver dans le répertoire `TMESolo_<nom>` et s'appeler `rapport.txt`
3. Lancez Mars et composez le TME solo en répondant aux questions ci-dessous. Enregistrez bien tous vos codes dans le répertoire `TMESolo_<nom>`.

Soumission de votre devoir à la fin du TME solo

1. Créez une archive contenant vos codes et votre rapport avec les commandes suivantes :
   `> cd`
   `> tar -cvf tmesolo_<nom>.tar TMESolo_<nom>/`
2. Déposez l'archive dans Moodle (dans la section rendu de TME il y a une remise de devoir intitulée TME solo). Attention vous devez valider votre soumission et vous ne pourrez soumettre qu'une seule fois.

Consigne importante : il vous est demandé de mettre des commentaires dans vos codes pour indiquer la correspondance entre les registres utilisés et les variables des programmes.

Le TME solo comporte une question principale sur 15 points (la note finale est sur 20) et une question pour atteindre 20 avec 5 points bonus.

Exercice 1 : Calcul de la validité d'un budget – 25 points

Question 1.1 : 15 points

Le programme donné ci-dessous teste, étant donné un tableau d'entiers naturels correspondants à des dépenses ou des recettes, si à la fin de ces opérations le bilan est bien nul et s'il n'y a jamais de passage à découvert. Une chaine est affichée pour indiquer le résultat de l'analyse via la fonction `bon_budget`.

Le tableau `recette_depense` ne contient que des valeurs positives, c'est la chaîne `chaine_rd` qui indique si une valeur est une dépense ou un recette. Ainsi si `chaine_rd[i]` vaut `'R'` alors `tab[i]` est une recette, sinon `chaine_rd[i]` vaut `'D'` et `tab[i]` est une dépense.

Écrivez dans un fichier nommé `Q1.s` (et placé dans le répertoire du TME solo) un programme assembleur correspondant au code C ci-dessous.

Testez votre programme. Le code C donne un exemple de base dont le bilan est juste, en commentaire les valeurs de `recette_depense` avec la même `chaine_rd` doivent faire afficher le contenu de la chaine `chaine_nok`.

N'oubliez pas de copier-coller votre programme assembleur dans votre rapport.

```
int recette_depense[] = {200, 150, 20, 30, 1100, 1000, 100}; // exemple ok
                        // 2ème exemple pas ok {300, 250, 50, 50, 1150, 1100, 100}
char chaine_rd[] = "RDDDRDD"; // chaine_rd[i] indique si l'élément
                              // recette_depense[i]  correspond à une
                              // recette (si 'R') ou une dépense (si 'D')
char ch_ok[]  = "bilan juste\n";
char ch_nok[] = "passage à decouvert ou mauvais calcul\n";

int bon_budget(int * t, char * s){
  int d = 0;
  int i = 0;

  while (s[i] != '\0'){
    if (s[i] == 'R') {   // cas recette, on ajoute
      d  = d + t[i];
    }
    else {               // cas dépense, on soustrait
      d = d - t[i];
    }
    if (d < 0){         // si valeur négative : découvert on s'arrête
      return d;
    }
    i++;
  }
  return d;            // on renvoie l'état final à la fin des opérations
}

void main(){
  int ok;
  ok = bon_budget(recette_depense, chaine_rd);
  if (ok != 0){
    printf("%s", ch_nok);
  }
  else{
    printf("%s", ch_ok);
  }
  exit();
}
```

Question 1.2 : 10 points

Copiez votre fichier contenant le code réponse de la question précédente en l'appelant `Q2.s` et ouvrez ce dernier dans Mars.

On souhaite programmer une version récursive de l'analyse de budget avec la fonction `bon_budget_rec` donnée ci-dessous. Le programme principal appelle désormais les 2 fonctions.

Le parcours se faisant (via les appels récursifs) dans le sens inverse, ce qui est interdit c'est d'avoir une balance positive en cours de route.

```
int bon_budget_rec(int * tab, char * ch, int index){
  int d = 0;
  if (ch[index] == '\0'){
    return 0;
  }
  // appel récursif sur la fin du tableau et de la chaine
  d = bon_budget_rec(tab, ch, index + 1);
  if (d > 0){ // état intermédiaire positif donc arrêt
    return d;
  }
  if (ch[index] == 'R'){
    return d + tab[index];  // cas recette
  }
  else {
    return d - tab[index];  // cas dépense
  }
}

void main(){
  int ok;
  ok = bon_budget(recette_depense, chaine_rd);
  if (ok != 0){
    printf("%s", ch_nok);
  }
  else{
    printf("%s", ch_ok);
  }
  ok = bon_budget_rec(recette_depense, chaine_rd, 0);
  if (ok == 0){
    printf("%s", ch_ok);
  }
  else{
    printf("%s",ch_nok);
  }
  exit();
}
```

Programmez la fonction `bon_budget_rec` (8 points).

Ensuite, modifiez le programme principal (2 points).

Testez votre programme pour vérifier qu'il fonctionne correctement sur les 2 exemples donnés (vous pouvez en inventer !).

N'oubliez pas de donner un minimum de commentaires.

Enfin, copiez votre programme réponse dans votre rapport.