Centre d'Intelligence Artificielle et de Robotique du Mali (CIAR-Mali)

Proposition de correction du contrôle n°3 d'Algorithmique et Programmation du 21 janvier 2026

Algorithmique et Programmation (MPSI 1ère année)

Questions du cours (5 points)

1. **B)** char nom[20];
2. **B)** '\0'
3. **B)** Un tableau de caractères se terminant par \0.
4. **C)** <string.h>
5. **A)** 4
6. **B)** Les fonctions comme printf ou strlen vont lire la mémoire au-delà de la chaîne jusqu'à trouver un \0 par hasard.
7. **B)** int n = c – '0';
8. **C)** Elle ajoute le contenu de src à la suite de dest.
9. **B)** Compare les adresses mémoire des deux tableaux.
10. **D)** strcmp()

Exercice 1 (5 points)

Vérification d'une chaîne correctement parenthésée

Astuce : On peut utiliser un compteur :

- qui sera incrémenté (+1) pour '('
- qui sera décrémenté (-1) pour ')'
- Si compteur devient négatif alors la chaîne n'est pas bien parenthésée. Cela signifie ')' vient avant '('
- Si compteur devient est plus grand que 1, alors la chaîne n'est pas bien parenthésée. Cela signifie qu'au moins un '(' n'est pas fermé.

- À la fin, si compteur = 0, alors la chaîne est bien parenthésée.

Algorithme

Fonction ParenthesesValides(c : chaine de caractères) : Booléen

```
  Variable
     i, compteur : entier
   compteur , i← 0
   Pour i de 0 jusqu'à longueur(chaine)-1 faire
         Si c[i] = '(' alors
             compteur ← compteur + 1
         Sinon si c[i] = ')' alors
             compteur ← compteur - 1
        FinSi
        Si compteur < 0 alors
             Retourner FAUX
        FinSi
   FinPour
   Si compteur = 0 alors
       Retourner VRAI
   Sinon
       Retourner FAUX
   FinSi
FinAlgorithme
```

Exercice 2 (5 points)

Astuce

1. Créer une nouvelle chaîne normalisée dans laquelle on supprime les espaces et ponctuations et toutes les lettres majuscules sont converties en minuscules.
2. Puis on vérifie si la chaîne est un palindrome

```
Fonction EstPalindromeNormalise(c : chaine de caractère) : Booléen
   // Normalisation
         i,j,debut,fin : entier
```

```
        s : chaine de caractère
    j ← 0
  Pour i de 0 jusqu'à longueur(chaine)-1 faire
// Si le caractère lu est minuscule, on le met dans la nouvelle chaine s
    Si ((c[i] >= 'a' et c[i] <='z') alors
      s[j] ← c[i]
      j ← j + 1
    FinSi
// Si le caractère lu est majuscule, on le convertit en minuscule puis on le met dans la nouvelle chaine s
        Si ((c[i] >= 'A' et c[i] <='Z') alors
            s[j] ← 'c[i]'+32 //Conversion de majuscule en minuscule
            j ← j + 1
        FinSi
  FinPour
  s[j] ← '\0'   //On met le caractère nul pour marquer la fin de la nouvelle chaine s

  // 2. Test palindrome
  debut ← 0 //l'indice à l'extrême gauche de la chaîne (ou début de la chaîne)
  fin ← j – 1 //l'indice à l'extrême droite de la chaîne (ou fin de la chaîne)
  Tant que (debut < fin) faire
    Si s[debut] ≠ s[fin] alors
      Retourner FAUX
    FinSi
    debut ← debut + 1
    fin ← fin - 1
  FinTantQue
  Retourner VRAI
FinAlgorithme
```

Exercice 3 (5 points)

Principe

- Ignorer espaces multiples
- Compter lorsqu'on passe d'un délimiteur à un caractère
- Afficher les mots

Délimiteurs : ' ', '\t', '\n', '\0'

```
void decouper(char chaine[]){
   int i =0 ; //Variable de parcours de la chaîne
   int nb_mots = 0 ; // Variable qui compte le nombre de mots
   int dans_mot = 0; // Variable qui renseigne qu'on a commencé la lecture d'un mot
   while (chaine[i] != '\0'){
      if (chaine[i] != ' ' && chaine[i] != '\t' && chaine[i] != '\n'){
//Si on commence la lecture de la chaîne et qu'on ne rencontre pas de délimitateur, on vérifie si on n'est pas déjà dans la lecture d'un mot
         if (!dans_mot){//Si on n'était pas dans un mot, alors, on vient de commencer un nouveau mot
            nb_mots++; // on augmente le nombre de mots
            printf("Mot %d : ", nb_mots);
            dans_mot = 1; //On signale qu'on est dans un mot
         }
//on affiche tous les caractères du mot dont on vient de commencer la lecture jusqu'à rencontrer un délimitateur
         printf("%c", chaine[i]);
      }
      else{
//Si on commence la lecture de la chaîne et qu'on rencontre un délimitateur, on vérifie si on est déjà dans la lecture d'un mot
         if (dans_mot){//Si on était déjà dans un mot et qu'on rencontre un délimitateur
            printf("\n"); //On fait un retour à la ligne
//la valeur de la variable dans_mot devient faux car on a vu un délimitateur
            dans_mot = 0;
         }
```

```
        }
        i++;
    }
//A la fin de la chaîne, on affiche le nombre de mots lus.
    printf("\n%d mots trouvés.\n", nb_mots);
}
```