**Les fonctions Correction**

**Exercice 1 : Fonction d'affichage**

**Écrivez une fonction appelée afficher\_message qui prend un message en tant qu'argument et l'affiche.**

#!/bin/bash

# Définition de la fonction

afficher\_message() {

message=$1

echo $message

}

# Appel de la fonction

afficher\_message "Bonjour, ceci est un message."

**Exercice 2 : Fonction de somme**

**Écrivez une fonction appelée somme qui prend deux nombres en tant qu'arguments et affiche leur somme.**

#!/bin/bash

# Définition de la fonction

somme() {

resultat=$(( $1 + $2 ))

echo "La somme de $1 et $2 est : $resultat"

}

# Appel de la fonction

somme 5 8

**Exercice 3 : Fonction de multiplication**

**Écrivez une fonction appelée produit qui prend deux nombres en tant qu'arguments et affiche leur produit.**

#!/bin/bash

# Définition de la fonction

produit() {

resultat=$(( $1 \* $2 ))

echo "Le produit de $1 et $2 est : $resultat"

}

# Appel de la fonction

produit 3 6

**Exercice 4 : Fonction avec valeur de retour**

**Écrivez une fonction appelée carré qui prend un nombre en tant qu'argument, calcule son carré, et renvoie le résultat.**

#!/bin/bash

# Définition de la fonction

carré() {

resultat=$(( $1 \* $1 ))

echo $resultat

}

# Appel de la fonction avec valeur de retour

resultat\_carré=$(carré 4)

echo "Le carré de 4 est : $resultat\_carré"

**Exercice 5 : Fonction récursive**

**Écrivez une fonction récursive appelée factorielle qui calcule la factorielle d'un nombre.**

#!/bin/bash

# Définition de la fonction récursive

factorielle() {

if [ $1 -eq 0 ]; then

echo 1

else

sous\_resultat=$(factorielle $(( $1 - 1 )))

echo $(( $1 \* sous\_resultat ))

fi

}

# Appel de la fonction récursive

resultat\_factorielle=$(factorielle 5)

echo "La factorielle de 5 est : $resultat\_factorielle"

**Exercice 6 : Fonction avec paramètres par défaut**

**Écrivez une fonction appelée salutation qui prend un nom en tant qu'argument et affiche une salutation. Si aucun nom n'est fourni, utilisez "Invité" par défaut.**

#!/bin/bash

# Définition de la fonction avec paramètre par défaut

salutation() {

nom=${1:-"Invité"}

echo "Bonjour, $nom !"

}

# Appel de la fonction

salutation "Alice"

salutation

**Exercice 7 : Fonction avec un nombre variable d'arguments**

**Écrivez une fonction appelée afficher\_elements qui prend un nombre variable d'arguments et les affiche un par un.**

#!/bin/bash

# Définition de la fonction avec un nombre variable d'arguments

afficher\_elements() {

for element in "$@"; do

echo $element

done

}

# Appel de la fonction

afficher\_elements "A" "B" "C" "D"

**Exercice 8 : Fonction avec tableau**

**Écrivez une fonction appelée afficher\_tableau qui prend un tableau en tant qu'argument et affiche chaque élément du tableau.**

#!/bin/bash

# Définition de la fonction avec tableau

afficher\_tableau() {

tableau=("$@")

for element in "${tableau[@]}"; do

echo $element

done

}

# Appel de la fonction

afficher\_tableau "1" "2" "3" "4"

**Exercice 9 : Fonction réutilisant une autre fonction**

**Écrivez une fonction appelée afficher\_message\_sans\_retour qui appelle la fonction afficher\_message (exercice 1) sans renvoyer de valeur.**

#!/bin/bash

# Définition de la fonction afficher\_message (exercice 1)

afficher\_message() {

message=$1

echo $message

}

# Définition de la fonction réutilisant afficher\_message

afficher\_message\_sans\_retour() {

afficher\_message "Ceci est un message sans retour."

}

# Appel de la fonction réutilisant afficher\_message

afficher\_message\_sans\_retour

**Exercice 10: Recherche de Fichiers Récursive**

**Écrivez un script Bash qui recherche de manière récursive tous les fichiers d'un répertoire donné qui ont une extension spécifique. Le script doit afficher le chemin complet de chaque fichier trouvé.**

**Ajoutez une fonctionnalité pour permettre à l'utilisateur de spécifier le répertoire de départ et l'extension de fichier lors de l'exécution du script en utilisant des options en ligne de commande.**

#!/bin/bash

# Fonction de recherche récursive des fichiers

recherche\_recursive() {

local repertoire=$1

local extension=$2

# Utilisation de find pour la recherche récursive

find "$repertoire" -type f -name "\*.$extension" 2>/dev/null

}

# Vérifier le nombre d'arguments

if [ "$#" -lt 2 ]; then

echo "Usage: $0 <répertoire> <extension>"

exit 1

fi

# Récupérer les arguments en ligne de commande

repertoire\_depart=$1

extension\_recherche=$2

# Vérifier si le répertoire existe

if [ ! -d "$repertoire\_depart" ]; then

echo "Erreur : Le répertoire '$repertoire\_depart' n'existe pas."

exit 1

fi

# Appeler la fonction de recherche récursive

resultats=$(recherche\_recursive "$repertoire\_depart" "$extension\_recherche")

# Vérifier si des fichiers ont été trouvés

if [ -n "$resultats" ]; then

echo "Fichiers trouvés :"

echo "$resultats"

else

echo "Aucun fichier trouvé avec l'extension '.$extension\_recherche'."

fi