Mini-calculatrice web

Dans ce TP, vous allez construire pas à pas une mini-application PHP permettant d'effectuer des opérations arithmétiques simples à partir d'un formulaire web. Vous mettrez en place le traitement des requêtes `POST`, la validation des données saisies et l'affichage des erreurs et du résultat.

question1) Créer un formulaire HTML `form` dans un fichier PHP principal contenant la structure HTML. Le formulaire doit envoyé une requête `POST` à la même page.

question2) Implémenter le traitement du formulaire : récupérer les valeurs `a`, `b` et `op` depuis `$_POST`. Vous pouvez vérifié en affichant temporairement leur contenu avec `var_dump()`.

question3) Mettre en place de la validation : refuser toute saisie vide ou non numérique et afficher un message d'erreur spécifique sous chaque champ (vérification coté PHP à la soumission) dans une balise `<small>`.

question4) Limiter les opérateurs possibles aux symboles `+`, `-`, `*`, `/` (vérification coté PHP). Une valeur non listée renvoie un message d'erreur visible "Opérateur invalide" dans une balise `<small>`.

question5) Implémenter le calcul lorsque les données sont valides. Afficher le résultat sous le formulaire dans une balise `<p>` avec la classe `result` au format `A op B = Résultat`.

question6) Utiliser la fonction `old` pour réafficher dans chaque champs l'ancienne valeur du calcul.

question7) Gérer le cas de la division par zéro en affichant une erreur claire sous le second champ (b). Une saisie `b=0` avec `/` ne renvoie aucun résultat et affiche "Division par zéro interdite ! ".

Statistiques sur une liste (PHP — formulaire, validation, affichage)

Dans ce TP, vous allez construire une mini-application web en PHP qui calcule des statistiques à partir d'une liste de nombres soumise via un formulaire. Vous mettrez en place le traitement serveur des données, la validation des entrées et l'affichage des résultats.

question1) Créer un formulaire `POST` avec un `<textarea name="list">` (plus un bouton de soumission) décrivant clairement le format attendu (séparateurs acceptés) et un emplacement d'erreur sous le champ.

question2) Implémenter une fonction de parsing normalisant les séparateurs (au minimum virgule, point-virgule et retours à la ligne), éliminant les entrées vides et rejetant toute valeur non numérique avec un message d'erreur sous le champ (dans une balise `small`).

question3) Calculer et afficher min, max, somme et moyenne à partir de la liste validée, puis présenter ces valeurs sous le formulaire dans un bloc dédié aux résultats (balise `div` de classe `result` pour le conteneur et des balises `p` pour chacun des résultats).

question4) Implémenter une fonction *médiane* qui trie la liste et retourne la valeur centrale (impair) ou la moyenne des deux centrales (pair). Par exemple, `list=1,2,3` affiche une médiane égale à `2` et que `list=1,2,3,4` affiche une médiane égale à `2.5`. Ajouter le resultat dans le bloc dédié aux résultats.

question5) Formater l'affichage d'une version ""nettoyée" de la liste (valeurs numériques déterministes, nombre d'éléments) et pour la précision des décimales (jusqu'à 6 décimales, sans zéros inutiles). Par exemple, une entrée `0.3333333333` s'affiche arrondie à `0.333333` dans la section de résultat (`number_format`).

Exploration de données : recherche sur un catalogue de dinosaures

Dans ce TP, vous allez construire une application PHP permettant de filtrer et d'afficher des données issues d'un fichier CSV. Vous mettrez en place la lecture des données, un formulaire GET complet de recherche, plusieurs filtres combinables, et une recherche sur texte via mot-clés. Le but est d'obtenir une page interactive capable d'afficher dynamiquement les résultats triés selon les critères choisis.

question1) Créer un formulaire `GET` avec plusieurs champs : recherche textuelle (`q`), catégorie (`carnivores/herbivores/tous`), ère (liste déroulante), bornes de poids (`wmin/wmax`), tri (`name/era/weight`) et limite de résultats. Chacun aura un `placeholder`, les données sur les catégories, ères viendront d'un tableau en php.

question2) Écrire une fonction de lecture du CSV qui lit le fichier `dinos.csv` et retourne un tableau associatif (une entrée par dinosaure).

question3) Implémenter la logique de filtrage côté serveur : exclure les lignes ne correspondant pas aux filtres actifs (catégorie, ère, bornes de poids). Vérifier avec `curl -S "http://localhost/TP_dino.php?category=carnivores"` que seuls les carnivores apparaissent dans la table.

question4) Utiliser la recherche floue sur le nom et les notes : utiliser la fonction `fuzzy_match_text()` et filtrer les entrées dont la similarité est inférieure à 0.35. Vérifier que la recherche `q=coorne` renvoie les 4 dinosaures : Carnotaurus, ceratosaurus, styracosaurus et triceratops.

question5) Mettre en place un tri dynamique selon le paramètre `sort`. Vérifier avec `curl -S "http://localhost/TP_dino.php?sort=` que la première ligne correspond au plus petit poids.

question6) Limiter le nombre de résultats avec le paramètre `limit` (valeur comprise entre 1 et 200) et ajouter une ligne indiquant le nombre de résultats globaux. Vérifier via `?limit=10` que seuls 10 résultats sont affichés.

question7) Afficher les résultats dans un tableau HTML contenant les colonnes Nom, Ère, Régime, Poids et Notes. Ajouter un message clair lorsqu'aucun résultat ne correspond aux filtres.

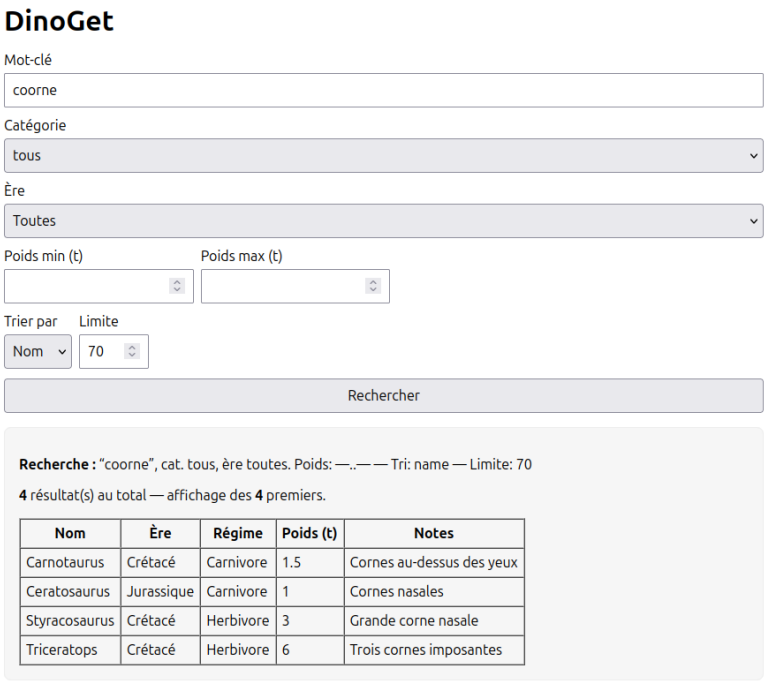

FIGURE 1 – Affichage attendu