Enrollment: 01-131232-004 CP Date: 29-10-2023
Name: Muhammad Jibran Assignment:1

Assignment # 1

Algorithm 1:

Overview:

1. Let consider starting location as point “A” and our destination as point “B”. Then find the displacement between them but using the given map.
2. As we have information about the pairs of locations, so we create list of distance of all possible paths.
3. Now compare each distance with the calculated displacement.
4. Select the path that have distance closest to displacement. **For example**, displacement=30km and there are three possible paths that is P1=33km, P2=35km and P3=37km. So P1 is closest to the 30km therefore we selected the P1.

Process:

1. Start.
2. First take the starting and destination location from the user. **(input)**
3. Store starting location in “A” and destination location in “B”.
4. Now find the displacement between “A” and “B” using map. **(processing)**
5. Generate a list that includes all possible routes and their corresponding distances between locations "A" and "B”.
6. Process all paths those join point “A” and “B”, make all possible routes by joining paths to reach the destination.
7. Now compare each route’s distance with displacement between “A” and “B”.
8. Select the route whose distance is closest to the displacement.
9. Output the selected route and the total distance. **(output)**
10. End.

Algorithm 2:

For sorting we have a lot of methos, some of them are:

- Bubble sorting.
- Insert sorting.
- Merge sorting.
- Quick sorting.
- Insertion sorting.
- Selection sorting.

Bubble sorting is an easy method, but it has more time complexity, so merging and quick sorting is the best option for large amounts of data. We use merge sorting.

Overview:

1. Take the number series.
2. Divide into two until get only one number left.
3. Start merging them and arranging them.
4. Sorting will end when we merge all divisions.

Process:

1. Start.
2. Take the series of numbers in array that need to be sorted. **(input)**

3. Divide into two halves (left half is first of series and right half is second part of series). **(processing)**
4. If there is an even number of integers then both divisions (first and second half) carry equal integers, but if series contain odd number of integers, then the second half contains one extra integer.
5. Then repeat the division process using a loop until you get the only one integer.
6. Now start merging them.
7. While merging two parts arrange them first then save them in array.
8. The list of integers will be sorted as all divided halves become merged.
9. Return sorted array. **(output)**
10. End.

Algorithm 3:

Overview:

1. Take n number.
2. Apply Fibonacci using temporary value storing method.
3. Return the value.

Process:

1. Start.
2. Take a number from the user and store in an integer n. **(input)**
3. For the loop take three integer n1=0, n2=1 and temp.
4. Use a loop and run it till n. **(processing)**
   - Temp=n1+n2.
   - n1=n2.
   - n2=temp.
5. Now print the value of temp if it is less than n.
6. Loop will print the Fibonacci sequence. **(output)**
7. End.

Algorithm 4:

Overview:

1. You have a database of store inventory.
2. Use loop to fulfil requirement.
3. Use the database and add, remove, and update the required item.
4. Generate the reports.

Process:

1. Initialize an empty inventory database, which can be represented as a dictionary or a similar data structure.
2. Start a loop that will continue until the user chooses to exit.
3. Inside the loop, display a menu with the following options:
   - 1."Add Item."
   - 2."Remove Item."
   - 3."Update Item Quantity."
   - 4."Generate Report."
   - 5."Exit"
4. Get the user's choice (an integer). **(input)**

5. If the user's choice is 1 (Add Item): **(process)**
    - Request the user for the item name.
    - Request the user for the item quantity.
    - Check if the item name already exists in the inventory.
    - If the item name exists, add the entered quantity to the existing quantity.
    - If the item name doesn't exist, create a new entry in the inventory with the item name and quantity.
6. If the user's choice is 2 (Remove Item): **(process)**
    - Request the user for the item name to remove.
    - Check if the item name is in the inventory.
    - If the item name exists, remove the item from the inventory.
    - If the item name doesn't exist, display "Item not found in inventory."
7. If the user's choice is 3 (Update Item Quantity): **(process)**
    - Request the user for the item name to update.
    - Request the user for the new quantity.
    - Check if the item name is in the inventory.
    - If the item name exists, update the quantity of the item to the new quantity.
    - If the item name doesn't exist, display "Item not found in inventory."
8. If the user's choice is 4 (Generate Report): **(process)**
    - Generate a report of the items and their quantities in the inventory.
    - Display the report to the user. **(output)**
9. If the user's choice is 5 (Exit): **(process)**
    - Exit the program by breaking out of the loop.
10. If the user's choice is not 1, 2, 3, 4, or 5, display "Invalid choice. Try again."
11. Repeat the loop from step 3 for further user interactions.
12. End the loop when the user chooses to exit (Option 5).
13. End.

Git-hub:

Link: https://github.com/NotDumbj/CP-Assingment-1.git

Group:

- Abdul Rafay.
- Muhammad Jibran.