CP :- ASSIGNMENT

Group TASK BY :-

Abdullah Nasir (01-131232-008)
Zainab Asif (01-131232-100)

Links :-

① Github.com/zainablone30/Zainab-Asif

② Github.com/itsmalikabdullahnasir/itsmalikabdullahnasir

"CP ASSIGNMENT:" = 01

★ Simple Algorithm to find the Shortest Path Between location:=

1. first, make a list of location (places) the distance between them, if you don't have one already.

2. ASK the user to tell you the starting location & the destination.

3. Start at the begining (the Starting location) & keep track of how far you've traveled, which is intially zero

4. Repeat the following step's until you reach your destination:=

   ⓐ Look at the places you can go to from where you are right now.

   ⓑ Choose the nearst place to move to.

   ⓒ Add the distance you traveled to the total distance.

5. Once you reach the destination you' ave found the shortest path & now far it is.

6. To find the exact path you took, follow the steps you took in reverse order from the destination back to the Starting location.

7. Tell the user shortest path & how far they needed to travel to get there.

∴ The alogorithm helps you find the quickest way to go from one place to another & tells you the total distance you'll travel.

TASK :- #02

∴ Sorting a list of Numbers :-

⇒ we will be keeping it simple as possible.

"NUMBER SORTING ALGORITHM"

① This Algorithm take's an array of un-sorted Number & sort's it out. (Input)

② The Algorithm take's the ~~first~~ very first two elements / Number of the array; Then Compare ~~compares~~ them with each other. ; To check which one is greater. (Processing)

③ The Algorithm then proceed ~~two~~ to the other unsorted element's. (Proccessing)

③ This is done until the end is reached ~~there's~~ ~~a~~ ~~so~~ there's a high chance that the array won't get sorted in an ascending order. So the algorithm will run again until the array get's sorted.

∴ Now we will see how it works.

① We will take an unsorted array of Number for etc :-

13	32	27	34	9

② Taking first two element's.

$\boxed{13 < 32}$

13	32	27	34	9

So; it is already sorted.

③ ~~[illegible]~~ Move on to the next element which is

$\boxed{32 > 27}$

13	27	32	34	9

It is in unsorted way so these must be swapped.

④ Now Compare Next two element :-

$\boxed{32 < 34}$

13	27	32	34	9

These are in sorted way so we will proceed.

⑤ Compare Next two value :-

$\boxed{34 > 9}$

13	27	32	9	34

So these need to be swapped.

∴ This Cycle need's to be carried out again until this become ture. (get's sorted)

9	13	27	32	34

~~[illegible]~~

Result!!

∴ This algorithm is simple & fast.
This is based upon Camparison algorithm method.

① Start

② Input (The Array)

③ Process (Start Comparing the element's)

④ Process (Continues the loop until true)

⑤ Print the output

⑥ End.

Question no.3

Algorithm for calculating fibonacci numbers

"The fibonacci sequence is a series of numbers, where each number is the sum of the two preceding ones (eg. 0,1,1,2,3,5,8,13,.....)."

1. Start with the input value 'n' representing the position of the desired fibonacci number.

2. Declare variables for two fibonacci numbers ie. 'a' and 'b'. Initialize 'a' to '0' and 'b' to '1'.

3. By using 'for' loop, calculate the fibonacci numbers from the 3rd position upto 'n'th position:

⇒ Repeat 'i' from '2' to 'n-1' since the loop already considers the first two fibonacci numbers

⇒ Calculate the next fibonacci number by declaring it a third varible 'c' as the sum of 'a' and 'b'.

⇒ Update 'a' to be the value of 'b'.

⇒ Update 'b' to be the value of 'c'.

4. After the loop, the 'n'th fibonacci number will be stored in either the 'a' or 'b' Variable, depending on whether 'n' is even or odd.

5. Return the value of 'n'th fibonacci number.

Question no. 4

Algorithm for inventory management.

1. Display a menu of options for the user.

⇒ Add and remove an item from inventory
⇒ Update quantities of existing items
⇒ Generate an inventory report.
⇒ Quit the program.

2. Execute the following steps according to the user's requirements:-

a. Add an item to Inventory:

⇒ Ask the user for the item name and quantity.

⇒ Check if the item name already exists in the inventory. If it does, update the quantity; otherwise, add another entry.

b. Remove an item from inventory:

⇒ Ask the user for item name and quantity.

⇒ Check if the item name exists in the inventory. If yes, remove the item from the inventory ; otherwise, display: "no items found."

c. Update the Quantity:
⇒ Ask the user for the item name and a new quantity.
⇒ Check if the item name exists in the inventory. If it does, update the quantity; otherwise, display: "no items found."

d. Generate Inventory Report:
⇒ ~~loop~~ Go through the inventory list and write down a report showing all the items and how many of each there are.

e. Quit the Program:
⇒ End the program.