CIS 121 Arrays and Vectors

1. Prompt the user for a number that will be used to dynamically define parallel arrays. One array will contain the make of an auto (Honda, Chevrolet etc) and the other array will contain the model (Accord etc).

   Once you have the number of the array, then allow the user to enter the data into each array using an loop.

   Write a function to display the arrays.

input	process	output
Number of autos (array size) Make of each auto (Honda, Chevy, Ford, etc.) Model of each auto (Accord, Malibu, etc.)	Prompt user for the number of cars. Create two dynamic parallel arrays of that size (one for make, one for model). Use a loop to store the make for each car into the first array. Use another loop to store the model for each car into the second array. Call a function and pass both arrays to it. Inside the function, use a loop to display the make and model paired by index.	A list of all cars showing Make and Model side-by-side.

2. Develop a structure that holds employee first name, employee last name, hours, rate and gross pay. Next create a vector of n occurrences of this structure. Loop over the structure to have the user enter data for each employee. Write a function that is called within this loop to compute gross pay and add it to the vector occurrence. Pass to this function the hours and rate and return gross pay. Give time and a half for hours over 40. Next, loop over the vectors of structures to display the structure elements. Display the employee's gross pay along with the other data within this loop.

input	process	output
Employee first name Employee last name Hours worked Rate of pay	Create a structure holding first name, last name, hours, rate, and gross pay.	Every employee's first name, last name, hours, rate, and gross pay.

Number of employees (vector size)	Ask the user for the number of employees. Resize a vector using this number. Loop and collect all employee data. Call a function for each employee to calculate gross pay. Gross pay = hours * rate Time and a half for hours > 40 Store the gross pay inside the structure. After the loop, use another loop to display all employee data including gross pay.	

3. Develop a structure that holds student first name, student last name, district code (I or O), enrolled credit hours and tuition balance. Make the structure a global structure by defining it outside of main and before any functions.

   Next, use this structure to create a dynamic vector. Write a loop that repeatedly prompts the user for first name, last name, district code, and credits taken. Use ctl+z to stop.

   For each user input, create a vector occurrence. Call a function to compute tuition. Pass to this function district code and enrolled credits and charge $250.00 per credit hour for in district students (district code == 'I') and 500.00 per credit hours for all other students. The function should compute and return tuition owed (enrolled credits x charge per credit hour). Load the tuition owed into the vector.

   Lastly, call another function to display each occurrence of the vector. Use a range-based loop to accomplish this. At the end display the number of students (number of vector occurrences) by using the size method of the vector.

input	process	output
Student first name Student last name District code (I or O) Enrolled credit hours Repeated entries loop	Create a structure holding name, district code, credits, and tuition owed. Create a vector. Use a loop that continues until the user enters c, t, or z to stop.	Full student list with name, district, credits, and tuition owed.

	For each student: Collect first and last name Collect district code Collect credit hours Call a function to compute tuition owed: If district == 'I' → charge 250 per credit If district == 'O' → charge 500 per credit Store the structure into the vector. After the loop, call another function to display all students. Use a range-based loop to output each structure.	