```
#include <iostream>

#include <string>

#include <vector>


//structs

struct Vector3D{

        float x;

        float y;

        float z;

};

struct Item{

        enum class Material { wood = 1, metal, plastic };

        std::string name;

        Material madeOf{ Material::wood };

        Vector3D dimensions{ 0,0,0 };

        int quantity{ 0 };

        float cost{ 0.0f };

};

struct Building{

        std::string name;

        Vector3D dimensions{ 0,0,0 };

        std::vector<Item> buildList;

        int lastID{ 0 };

};

//addItem function

Item addItem() {

        Item item;

        std::cout << "Enter the name of the item: " << std::endl;

        std::cin >> item.name;
```

```
std::cout << "Enter the quantity: " << std::endl;
std::cin >> item.quantity;
std::cout << "Enter the cost: " << std::endl;
std::cin >> item.cost;
std::cout << "Enter the dimensions of the item: " << std::endl;
std::cout << "Enter x: " << std::endl;
std::cin >> item.dimensions.x;
std::cout << "Enter y: " << std::endl;
std::cin >> item.dimensions.y;
std::cout << "Enter z: " << std::endl;
std::cin >> item.dimensions.z;
std::cout << "Enter what material the item is (1 = wood, 2 = metal, 3 = plastic): " << std::endl;

int materialChoice;
std::cin >> materialChoice;
while (materialChoice < 1 || materialChoice > 3) {
        std::cout << "Please enter a correct material choice (1-3): " << std::endl;
        std::cin >> materialChoice;
}

switch (materialChoice) {
case 1:
        item.madeOf = Item::Material::wood;
        break;
case 2:
        item.madeOf = Item::Material::metal;
        break;
case 3:
        item.madeOf = Item::Material::plastic;
```

```
			break;
		}
		return item;
}
//add a new building function
Building newBuilding(int& lastID) {
		Building addBuilding;
		std::cout << "Would you like to add a building? 1 for Yes, 0 for no" << std::endl;
		int buildChoice;
		lastID++;
		addBuilding.lastID = lastID;
		std::cin >> buildChoice;

		if (buildChoice == 1) {
			std::cout << "Enter the name of the building: " << std::endl;
			std::cin >> addBuilding.name;
			std::cout << "Enter the dimensions of the Building (enter x, then y, then z.): " << 
std::endl;
			std::cout << "Enter x: " << std::endl;
			std::cin >> addBuilding.dimensions.x;
			std::cout << "Enter y: " << std::endl;
			std::cin >> addBuilding.dimensions.y;
			std::cout << "Enter z: " << std::endl;
			std::cin >> addBuilding.dimensions.z;
			std::cout << "Do you want to add an item to the building? 1 for yes, 0 for no:  " << 
std::endl;
			int itemChoice;
			std::cin >> itemChoice;
			if (itemChoice == 1) {
				addBuilding.buildList.push_back(addItem());
			}
			else if (itemChoice == 0) {
				std::cout << "Done Entering Items." << std::endl;
			}
			return addBuilding;
		}
		else if (buildChoice == 0) {
			std::cout << "Done Entering Buildings." << std::endl;
		}
}
//print the buildlist function
void printBuildList(const Building& currBuilding) {
		std::cout << "Building Name: " << currBuilding.name << std::endl;
		std::cout << "x dimension of building: " << currBuilding.dimensions.x << " inches" << std::endl;
		std::cout << "y dimension of building: " << currBuilding.dimensions.y << " inches" << std::endl;
		std::cout << "z dimension of building: " << currBuilding.dimensions.z << " inches" << std::endl;

		for (int i{ 0 }; i < currBuilding.buildList.size(); i++) {
			std::cout << "Item name: " << currBuilding.buildList[i].name << std::endl;
			std::cout << "Item cost:  " << currBuilding.buildList[i].cost << std::endl;
			std::cout << "x dimension: " << currBuilding.buildList[i].dimensions.x << std::endl;
			std::cout << "y dimension: " << currBuilding.buildList[i].dimensions.y << std::endl;
			std::cout << "z dimension: " << currBuilding.buildList[i].dimensions.z << std::endl;
			std::cout << "Item quantity:  " << currBuilding.buildList[i].quantity << std::endl;

			switch (currBuilding.buildList[i].madeOf) {
			case Item::Material::metal:
				std::cout << "Made of Metal" << std::endl;
```

```
                break;
            case Item::Material::wood:
                std::cout << "Made of Wood" << std::endl;
                break;
            case Item::Material::plastic:
                std::cout << "Made of Plastic" << std::endl;
                break;
            }
            std::cout << std::endl;
    }
}
//finds the building through indexing function
int findBuilding(int ID, const std::vector<Building>& blueprints) {
    int i{ 0 };
    for (; i < blueprints.size() && blueprints[i].lastID != ID; i++) {} //takes ID of building, uses linear
search through vector to find the index that matches
    if (i < blueprints.size()){
            return i;
    }
    std::cout << "could not find blueprint" << std::endl;
    return -1;
}


int main() {
    int id;
    int index;
    int ID{ 0 };
    bool exit{ false };
    int choice{ 0 };
```

```
	std::vector<Building> blueprints;

	//choose an option while loop

	while (exit == false) {

		std::cout << "Please choose an option" << std::endl;

		std::cout << "Press (1) to Add Building " << std::endl;

		std::cout << "Press (2) to Print Blueprint " << std::endl;

		std::cout << "Press (3) to Exit Program " << std::endl;

		int choice{ 0 };


		std::cin >> choice;


		switch (choice) {

//calls the different functions based on what you want to do

		case 1:

			blueprints.push_back(newBuilding(ID));

			break;

		case 2:

			std::cout << std::endl << "Enter id of the building: ";

			std::cin >> id;

			index = findBuilding(id, blueprints);

			if (index != -1) {

				printBuildList(blueprints[index]);

			}

			else {

				std::cout << std::endl << id << " Not found in list." << std::endl;

			}

			break;

		case 3:

			exit = true;
```

```
                break;
            }

        }
        return 0;
    }
```