Objective:

- Struct Object as attribute of another struct
- Array of Struct Objects
- Use of Enumeration and Union

Task-1:
The example, which we discussed today in class.

```
struct Address
{
    char city[30];
    char country[50];
    int streetNo;
    char block[30];
    char colony[100];
};
```

```
struct Phone
{
    int intlCode;
    int cityCode;
    int phoneNo;
};
```

```
struct Student
{
    char rollNo[15];
    char name[40];
    Address ads;
    Phone ph;
};
```

```
int main()
{
    Student std[10];
    std[0].ph.cityCode = 42;
    std[0].ph.intlCode = 92;
    std[0].ph.phoneNo = 385679;

    //you can also do something like this
    Address ad;
    strcpy(ad.city,"Lahore"); //You should use here stringCopy function given in
                              //MyOOPString.h practice/header file in your PF
    strcpy(ad.country,"Pakistan");
    ad.streetNo=23;
    strcpy(ad.block,"West B");
    strcpy(ad.colony,"Mars");

    std[0].ads = ad; //shallow copy safe for address object

    strcpy(std[0].name,"Ahmed");
    strcpy(std[0].rollNo,"BCSF01M001");
    return 0;
}
```

Task-2: ***Drink Machine Simulator***
Write a program that simulates a soft drink machine. The program should use a structure that stores the following data:

Drink Name
Drink Cost
Number of Drinks in Machine

The program should create an array of five structure objects. The elements should be initialized with the following data:

Drink Name	Cost	Number in Machine
Cola	.75	20
Root Beer	.75	20
Lemon-Lime	.75	20
Grape Soda	.80	20
Cream Soda	.80	20

Each time the program runs, it should enter a loop that performs the following steps: A list of drinks is

displayed on the screen. The user should be allowed to either quit the program or pick a drink. If the user selects a drink, he or she will next enter the amount of money that is to be inserted into the drink machine. The program should display the amount of change that would be returned and subtract one from the number of that drink left in the machine. If the user selects a drink that has sold out, a message should be displayed. The loop then repeats. When the user chooses to quit the program, it should display the total amount of money the machine earned.

Input Validation: When the user enters an amount of money, do not accept negative values, or values greater than $1.00.

Task-3: ***Enumeration***

The purpose is to increase the code readability.
Comment and discuss: Why 3rd approach is better than 2nd.

Poor Approach	Better	Best

int main()
{
int color;
cin>>color;
if (color==0)//RED color
{
//...your code
}
else if (color==1)
//Green color
{
//...your code
}
return 0;
}
	int main()
	{
	int color;
	cin>>color;
	const int RED=0, GREEN=1;
	if (color==RED)
	{
	//...your code
	}
	else if (color==GREEN)
	{
	//...your code
	}
	return 0;
	}
		enum Colors {RED, GREEN, BLUE};
		int main()
		{
		int color;
		cin>>color;
		if (color==RED)
		{
		//...your code
		}
		else if (color == Colors::GREEN)
		{
		//...your code
		}
		return 0;
		}

Task-4: ***union: memory saver – but not just a memory saver***

Have a critique look at the following snippet:

Example with Union	Example without Union

union ABC
{
int a;
double b;
char c[5];
};
int main()
{
cout<<sizeof(ABC)<<endl;
ABC obj;
//size of obj is 8B
//now wll treat it as int
obj.a=65;//writes at first 4B of obj
cout<<obj.a<<endl;//reads from first 4B of obj
//65 will be stored at first byte of obj
//Can i treat it as char? Yes
cout<<obj.c[0]<<endl;
return 0;
}
	int main()
	{
	//such behavior maybe simulated in absence of union but have to deal with typecasting our self.
	double sim;//same memory size as ABC
	//treat sim as int
	(int)(&sim) = 65;
	cout<<*(int*)&sim<<endl;
	//now treat as char
	cout<<((char*)&sim)[0]<<endl;
	return 0;
	}

Both snippets produce same output and consume exactly the same amount of memory	

Explore the following link for miscellaneous reading about union:
https://www.geeksforgeeks.org/difference-structure-union-c/