Lab no 11: Program to use the nested structure.

```
#include <stdio.h>

// Nested structure definition
struct City {
    char cityName[50];
    int zipCode;
};

struct State {
    char stateName[50];
    char stateCode[3];
};

struct Address {
    struct City city;
    struct State state;
};

int main() {
    // Creating an instance of the nested structure
    struct Address myAddress;

    // Assigning values to the nested structures
    printf("Enter city name: ");
    scanf("%s", myAddress.city.cityName);
    printf("Enter zip code: ");
    scanf("%d", &myAddress.city.zipCode);

    printf("Enter state name: ");
    scanf("%s", myAddress.state.stateName);
    printf("Enter state code: ");
    scanf("%s", myAddress.state.stateCode);

    // Displaying the entered information
    printf("\nYour address:\n");
    printf("City: %s, Zip Code: %d\n", myAddress.city.cityName, myAddress.city.zipCode);
    printf("State: %s, State Code: %s\n", myAddress.state.stateName, myAddress.state.stateCode);

    return 0;
}
```

Output:

```
Enter city name: Pokhara
Enter zip code: 33100
Enter state name: Gandaki
Enter state code: 04

Your address:
City: Pokhara, Zip Code: 33100
State: Gandaki, State Code: 04
```

Scenarios where structures and unions are commonly used in real-world programming:

Structures:

1. **Database Modeling:** In database systems, structures can represent the structure of a record or a table. Each field in the record can be represented by a member of the structure.
2. **Graphics Programming:** In graphics programming, structures are often used to represent geometric shapes or objects, storing information such as coordinates, colors, and sizes.
3. **File Handling:** Structures are employed to define the structure of data stored in files. Each record in a file can be represented by a structure.
4. **Networking:** In networking applications, structures can be used to define packet formats. Each field in the packet can be represented by a member of the structure.

Unions:

1. **Memory Optimization:** Unions are useful when you want to represent a data structure that can be interpreted in different ways. It allows multiple members to share the same memory location, saving space.
2. **Hardware Interaction:** Unions can be used when interacting with hardware, where a specific memory layout is required for communication.
3. **Parsing Data:** When parsing different types of data (such as in network protocols or file formats), unions can be used to interpret the same memory space differently based on the context.
4. **Configuration Settings:** Unions can be employed to represent a configuration setting that can be of different types, allowing flexibility in handling various data types within a fixed memory space.

The use of structures and unions depends on the specific requirements of the program or system you are working on. Structures are typically used when you want to group related data, while unions are used when you need to share memory among different data types.