GODADDY

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>

#define MAX_DOMAINS 100
#define MAX_HOSTINGS 100
#define MAX_USERS 100

typedef struct {
    int user_id;
    char username[50];
    char email[100];
} User;

typedef struct {
    int domain_id;
    char domain_name[100];
    int user_id;
    char registration_date[11];
    char status[10];
} Domain;
```

```
typedef struct {
   int hosting_id;
   int user_id;
   char plan_type[50];
   char renewal_date[11];
} Hosting;

typedef struct {
   int password_id;
   int user_id;
   int two_factor_auth;
   int ssl_certification;
   int malware_scanning;
} Security;

Domain domains[MAX_DOMAINS];
Hosting hostings[MAX_HOSTINGS];
Security securities[MAX_USERS];
User  users[MAX_USERS];

int domain_count = 0;
int hosting_count = 0;
int user_count = 0;

bool is_valid_date(const char* date) {
```

```
    // Simple date validation (YYYY-MM-DD)
    if (strlen(date) != 10 || date[4] != '-' || date[7] != '-') {
        return false;
    }
    return true;
}

void register_user(const char* username, const char* email) {
    if (user_count < MAX_USERS) {
        users[user_count].user_id = user_count + 1;
        strcpy(users[user_count].username, username);
        strcpy(users[user_count].email, email);
        user_count++;
        printf("User  registered: %s\n", username);
    } else {
        printf("User  limit reached.\n");
    }
}

void register_domain(int user_id, const char* domain_name, const char* registration_date) {
    if (domain_count >= MAX_DOMAINS) {
        printf("Domain limit reached.\n");
        return;
    }
```

```
    if (!is_valid_date(registration_date)) {
        printf("Invalid registration date format. Use YYYY-MM-DD.\n");
        return;
    }

    // Check if the domain already exists
    for (int i = 0; i < domain_count; i++) {
        if (strcmp(domains[i].domain_name, domain_name) == 0) {
            printf("Domain already registered: %s\n", domain_name);
            return;
        }
    }

    domains[domain_count].domain_id = domain_count + 1;
    strcpy(domains[domain_count].domain_name, domain_name);
    domains[domain_count].user_id = user_id;
    strcpy(domains[domain_count].registration_date, registration_date);
    strcpy(domains[domain_count].status, "active");
    domain_count++;
    printf("Domain registered: %s\n", domain_name);
}

void add_hosting_plan(int user_id, const char* plan_type, const char* renewal_date) {
    if (hosting_count < MAX_HOSTINGS) {
        hostings[hosting_count].hosting_id = hosting_count + 1;
```

```
        hostings[hosting_count].user_id = user_id;
        strcpy(hostings[hosting_count].plan_type, plan_type);
        strcpy(hostings[hosting_count].renewal_date, renewal_date);
        hosting_count++;
        printf("Hosting plan added: %s\n", plan_type);
    } else {
        printf("Hosting limit reached.\n");
    }
}

void add_security_features(int user_id, int two_factor_auth, int ssl_certification, int malware_scanning)
{
    if (user_count < MAX_USERS) {
        securities[user_count].password_id = user_count + 1;
        securities[user_count].user_id = user_id;
        securities[user_count].two_factor_auth = two_factor_auth;
        securities[user_count].ssl_certification = ssl_certification;
        securities[user_count].malware_scanning = malware_scanning;
        user_count++;
        printf("Security features added for user ID: %d\n", user_id);
    } else {
        printf("User  limit reached.\n");
    }
}

void display_domains() {
```

```
    printf("\nRegistered Domains:\n");
    for (int i = 0; i < domain_count; i++) {
        printf("ID: %d, Name: %s, User ID: %d, Registration Date: %s, Status: %s\n",
               domains[i].domain_id, domains[i].domain_name, domains[i].user_id,
               domains[i].registration_date, domains[i].status);
    }
}

void display_hostings() {
    printf("\nHosting Plans:\n");
    for (int i = 0; i < hosting_count; i++) {
        printf("ID: %d, User ID: %d, Plan Type: %s, Renewal Date: %s\n",
               hostings[i].hosting_id, hostings[i].user_id,
               hostings[i].plan_type, hostings[i].renewal_date);
    }
}

void display_users() {
    printf("\nRegistered Users:\n");
    for (int i = 0; i < user_count; i++) {
        printf("User  ID: %d, Username: %s, Email: %s\n",
               users[i].user_id, users[i].username, users[i].email);
    }
}
```

```
int main() {
    int ch;
    char username[50], email[100];
    Domain a;
    Hosting b;
    Security c;

    while (1) {
        printf("\n1. Register User\n2. Register Domain\n3. Add Hosting Plan\n4. Add Security Features\n5. Display Domains\n6. Display Hosting Plans\n7. Display Users\n8. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &ch);

        switch (ch) {
            case 1:
                printf("Enter Username and Email: ");
                scanf("%s %s", username, email);
                register_user(username, email);
                break;

            case 2:
                printf("Enter User ID, Domain Name, Registration Date (YYYY-MM-DD): ");
                scanf("%d %s %s", &a.user_id, a.domain_name, a.registration_date);
                register_domain(a.user_id, a.domain_name, a.registration_date);
                break;
```

```
        case 3:
            printf("Enter User ID, Plan Type, Renewal Date (YYYY-MM-DD): ");
            scanf("%d %s %s", &b.user_id, b.plan_type, b.renewal_date);
            add_hosting_plan(b.user_id, b.plan_type, b.renewal_date);
            break;

        case 4:
            printf("Enter User ID, Two Factor Auth (1/0), SSL Certification (1/0), Malware Scanning (1/0): ");
            scanf("%d %d %d %d", &c.user_id, &c.two_factor_auth, &c.ssl_certification,
&c.malware_scanning);
            add_security_features(c.user_id, c.two_factor_auth, c.ssl_certification, c.malware_scanning);
            break;

        case 5:
            display_domains();
            break;

        case 6:
            display_hostings();
            break;

        case 7:
            display_users();
            break;

        case 8:
```

```
            exit(0);
            default:
            printf("invalid choice");
        }
    }
}
```