Blood Bank Management System in C

Context

Assume you are tasked with developing a Blood Bank Management System for a local hospital. The system should manage the blood groups available in the blood bank. The project should include functionalities such as adding, deleting, searching, and managing blood group packages. This system will help the hospital efficiently manage their blood stock and ensure they have the necessary blood types available when needed.

Task

Develop a C program to implement a Blood Bank Management System. Your program should include separate functions for adding blood groups, deleting blood groups, searching blood groups, creating blood group packages, and other relevant operations. The project should utilize concepts such as if-else statements, loops, 1D and 2D arrays, functions, strings, structures, and pointers.

Project Requirements:

1. **Structures**:
    - Use structures to represent blood groups. Each structure should contain the blood type (e.g., A+, B-, O+), the quantity available, and any other relevant information.
    - Use 1D and 2D arrays to store and manage the blood group data.
2. **Functions**:
    - **Add Blood Group**: A function to add a new blood group to the system.
    - **Delete Blood Group**: A function to delete an existing blood group from the system.
    - **Search Blood Group**: A function to search for a blood group by its type.
    - **Blood Group Package**: A function to create a package containing multiple units of different blood groups.
    - **Display Blood Groups**: A function to display all available blood groups and their quantities.
    - **Update Blood Quantity**: A function to update the quantity of a specific blood group.
3. **Programming Concepts**:
    - Use if-else statements to handle different cases, such as checking if a blood group already exists before adding it.
    - Use loops to iterate through arrays and perform operations such as searching and displaying blood groups.
    - Use functions to modularize the code and perform specific tasks.
    - Use strings to handle blood type names.
    - Use pointers to manipulate data and interact with arrays and structures.

Detailed Requirements:

1. **Structure Definition**: Define a structure for blood groups with fields for blood type (string), quantity (integer), and any other relevant information.
2. **Function Implementations**:
   - *void addBloodGroup(struct BloodGroup *bloodGroups, int *count);*
   - *void deleteBloodGroup(struct BloodGroup *bloodGroups, int *count, const char *bloodType);*
   - *int searchBloodGroup(struct BloodGroup *bloodGroups, int count, const char *bloodType);*
   - *void createBloodGroupPackage(struct BloodGroup *bloodGroups, int count, const char *packageDetails);*
   - *void displayBloodGroups(struct BloodGroup *bloodGroups, int count);*
   - *void updateBloodQuantity(struct BloodGroup *bloodGroups, int count, const char *bloodType, int quantity);*
3. **Exception Handling**: Handle cases where the blood group array is full or empty, or when a specified blood group does not exist.
4. **User Interaction**: Create a simple text-based menu for users to interact with the system. The menu should provide options to add, delete, search, display, and update blood groups, as well as to create blood group packages.
5. **Example Main Function**:

```
int main() {
    struct BloodGroup bloodGroups[MAX_BLOOD_GROUPS];
    int count = 0;
    int choice;
    char bloodType[10];
    int quantity;

    while (1) {
        printf("\nBlood Bank Management System\n");
        printf("1. Add Blood Group\n");
        printf("2. Delete Blood Group\n");
        printf("3. Search Blood Group\n");
        printf("4. Display Blood Groups\n");
        printf("5. Update Blood Quantity\n");
        printf("6. Create Blood Group Package\n");
        printf("7. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                addBloodGroup(bloodGroups, &count);
                break;
            case 2:
                printf("Enter blood type to delete: ");
                scanf("%s", bloodType);
                deleteBloodGroup(bloodGroups, &count, bloodType);
                break;
            case 3:
                printf("Enter blood type to search: ");
                scanf("%s", bloodType);
                if (searchBloodGroup(bloodGroups, count, bloodType) != -1) {
                    printf("Blood group found.\n");
                } else {
                    printf("Blood group not found.\n");
                }
                break;
            case 4:
                displayBloodGroups(bloodGroups, count);
                break;
            case 5:
                printf("Enter blood type to update: ");
                scanf("%s", bloodType);
                printf("Enter new quantity: ");
                scanf("%d", &quantity);
                updateBloodQuantity(bloodGroups, count, bloodType, quantity);
                break;
            case 6:
                createBloodGroupPackage(bloodGroups, count, "PackageDetails");
                break;
            case 7:
                exit(0);
            default:
                printf("Invalid choice!\n");
        }
    }
    return 0;
}
```

Deliverables:

Source Code: Complete C program with clearly defined functions.
Documentation: A report explaining the functions implemented, and how the system works.
Output File: Sample output demonstrating the functionality of the Blood Bank Management System.