Assignment Question 9: Smart Irrigation System Optimizer

Scenario: You're designing a Smart Irrigation System Optimizer to manage farm components (e.g., "Pump", "Pipe", "Valve", "Sensor", "Sprinkler") for precision agriculture. The system uses:

- **Water Request System (Queue):** Irrigation requests queue up from soil sensors.
- **Emergency Watering (Stack):** Drought alerts stack stack in LIFO order for immediate watering.
- **Water Usage Log (Array):** Water usage logs into an array-based farm record (size: 5 slots). If full, the oldest is archived.
- **Maintenance Tracker (Linked Lists):**
    - Leaky components go to a singly linked list.
    - Fixed components move to a doubly linked list for review.
    - High-priority components cycle in a circular linked list for urgent fixes.

Objective: Simulate irrigation management, logging, and component maintenance.

Tasks:

a) **Request and Emergency**

- Simulate 6 requests (e.g., "Pump", "Pipe", "Valve", "Sensor", "Sprinkler", "Filter") arriving in a queue.
- Stack drought alerts in LIFO order. Write pseudocode or an algorithm to:
    - Enqueue all 6 requests.
    - Dequeue and push onto a stack.
    - Pop to show watering order.
- *Creativity Bonus:* Describe (in 2-3 sentences) why LIFO suits (e.g., "Filter" last to ensure clean water flow).

b) **Water Usage Log**

- Log usage in a 5-slot array.
- Simulate logging 7 usages (e.g., "Use1", "Use2", ..., "Use7"). If full, archive the oldest. Write pseudocode or an algorithm to:
    - Insert the first 5 usages.
    - Handle overflow for "Use6" and "Use7".
- *Creativity Bonus:* Suggest (in 2-3 sentences) a reason for archiving (e.g., seasonal water budgeting).

c) **Leaky Component Tracker**

- "Pipe" and "Sprinkler" are leaky. Add to a singly linked list.
- Move "Pipe" to a doubly linked list post-fix. Write pseudocode or an algorithm to:

    - Insert "Pipe" and "Sprinkler".
    - Delete "Pipe" and insert it into the doubly linked list.
    - Traverse forward and backward.
- *Creativity Bonus:* Propose (in 2-3 sentences) a leak cause and fix (e.g., "Pipe cracked by frost, sealed with resin").

d) **Priority Fixes**

- "Pump" and "Valve" need urgent fixes (e.g., pressure loss). Add to a circular linked list. Write pseudocode or an algorithm to:
    - Insert "Pump" and "Valve".
    - Traverse twice.
- *Creativity Bonus:* Invent (in 2-3 sentences) a fix tweak (e.g., "Valve gets a smart pressure gauge").

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>


#define MAX_LOGS 5 // max entries in usage log


// ----------- structures ----------- //

Typedef struct Node {

   Char name[20];

   Struct Node* next;

} Node;


Typedef struct DNode {

   Char name[20];

   Struct DNode* prev;

   Struct DNode* next;
```

```
} DNode;

Typedef struct CNode {
    Char name[20];
    Struct CNode* next;
} CNode;

// ----------- globals ----------- //
Char* reqQueue[10];
Int qFront = -1, qRear = -1;

Char* droughtStack[10];
Int sTop = -1;

Char* usageLog[MAX_LOGS];
Int logCount = 0;

Node* leakyList = NULL;
DNode* fixedList = NULL;
CNode* urgentList = NULL;

// ----------- queue part ----------- //
Void pushToQueue(char* comp) {
    If (qRear < 9) {
        If (qFront == -1) qFront = 0;
        reqQueue[++qRear] = comp;
```

```
    }
}

Char* popFromQueue() {
    If (qFront == -1 || qFront > qRear) return NULL;
    Return reqQueue[qFront++];
}

// ----------- stack part ----------- //
Void addEmergency(char* comp) {
    If (sTop < 9) {
        droughtStack[++sTop] = comp;
    }
}

Char* handleEmergency() {
    If (sTop == -1) return NULL;
    Return droughtStack[sTop--];
}

// ----------- logging part ----------- //
Void addToUsageLog(char* entry) {
    If (logCount < MAX_LOGS) {
        usageLog[logCount++] = entry;
    } else {
        // pushing out oldest usage
```

```
        For (int I = 1; I < MAX_LOGS; i++) {
            usageLog[I – 1] = usageLog[i];
        }
        usageLog[MAX_LOGS – 1] = entry;
    }
}

Void showUsageLogs() {
    Printf("\n>>> Water Usage Log:\n");
    For (int I = 0; I < logCount; i++) {
        Printf("  - %s\n", usageLog[i]);
    }
}

// ---------- maintenance stuff ---------- //
Void addLeaky(char* comp) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    Strcpy(newNode->name, comp);
    newNode->next = leakyList;
    leakyList = newNode;
}

Void fixComponent(char* comp) {
    // take out from leaky and move to fixed
    Node *curr = leakyList, *prev = NULL;
    While (curr != NULL && strcmp(curr->name, comp) != 0) {
```

```
        Prev = curr;
        Curr = curr->next;
    }

    If (curr != NULL) {
        If (prev == NULL)
            leakyList = curr->next;
        else
            prev->next = curr->next;
        free(curr);

        DNode* fixedNode = (DNode*)malloc(sizeof(DNode));
        Strcpy(fixedNode->name, comp);
        fixedNode->prev = NULL;
        fixedNode->next = fixedList;
        if (fixedList) fixedList->prev = fixedNode;
        fixedList = fixedNode;
    }
}

Void showFixedForward() {
    DNode* ptr = fixedList;
    Printf("\n>>> Fixed Components (forward):\n");
    While (ptr) {
        Printf("  - %s\n", ptr->name);
        Ptr = ptr->next;
```

```
    }
}

Void showFixedBackward() {
    DNode* ptr = fixedList;
    If (!ptr) return;
    While (ptr->next) ptr = ptr->next;

    Printf(“>>> Fixed Components (backward):\n”);
    While (ptr) {
        Printf(“  - %s\n”, ptr->name);
        Ptr = ptr->prev;
    }
}

// ----------- circular urgent list ----------- //
Void addUrgent(char* comp) {
    CNode* newNode = (CNode*)malloc(sizeof(CNode));
    Strcpy(newNode->name, comp);

    If (!urgentList) {
        urgentList = newNode;
        newNode->next = urgentList;
    } else {
        CNode* temp = urgentList;
        While (temp->next != urgentList)
```

```
            Temp = temp->next;
        Temp->next = newNode;
        newNode->next = urgentList;
    }
}

Void cycleUrgentTwice() {
    If (!urgentList) return;
    CNode* temp = urgentList;
    Int rounds = 0;

    Printf("\n>>> Urgent Fix Components (cycled twice):\n");
    Do {
        Printf("  - %s\n", temp->name);
        Temp = temp->next;
        If (temp == urgentList) rounds++;
    } while (rounds < 2);
}

// ----------- main logic ----------- //
Int main() {
    Printf("=== Smart Irrigation System Simulation ===\n");

    // --- part a: requests and drought emergencies --- //
    Char* comps[] = {"Pump", "Pipe", "Valve", "Sensor", "Sprinkler", "Filter"};
    For (int I = 0; I < 6; i++) pushToQueue(comps[i]);
```

```
Char* out;
While ((out = popFromQueue()) != NULL) {
    addEmergency(out);
}

Printf("\n>>> Watering Order (LIFO – Emergency):\n");
While ((out = handleEmergency()) != NULL) {
    Printf("  - Water: %s\n", out);
}
// LIFO is handy here 'cause newest drought spots or filters might be more critical.

// --- part b: usage logging --- //
Char* logs[] = {"Use1", "Use2", "Use3", "Use4", "Use5", "Use6", "Use7"};
For (int I = 0; I < 7; i++) addToUsageLog(logs[i]);
showUsageLogs();
// older logs get replaced when space runs out – helps avoid memory clutter and track recent activity only.

// --- part c: maintenance tracker --- //
addLeaky("Pipe");
addLeaky("Sprinkler");

fixComponent("Pipe"); // Pipe got fixed
showFixedForward();
showFixedBackward();
// Pipe leak probably from pressure or cold snap — fixed with sealant.
```

```
    // --- part d: urgent circular list --- //
    addUrgent(“Pump”);
    addUrgent(“Valve”);
    cycleUrgentTwice();
    // Valve now upgraded with a smart pressure reader — can auto-adjust!

    // TODO: maybe add user input version later?

    return 0;
}
```