**ASSIGNMENT**

**NAME:P.PAVITHRA**

**ROLL NO:ME24B1031**

**DATE:**

**QUESTION NO : 9**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX 6

// Queue structure

typedef struct

{

char\* items[MAX];

int front, rear;

} Queue;

// Stack structure

typedef struct

{

char\* items[MAX];

int top;

} Stack;

// Singly Linked List Node

typedef struct Node

{

char\* data;

struct Node\* next;

} Node;

// Doubly Linked List Node

typedef struct dNode

{

char\* data;

struct dNode\* next;

struct dNode\* prev;

} dNode;

// Circular Linked List Node

typedef struct cNode

{

char\* data;

struct cNode\* next;

} cNode;

// Queue functions

void createQueue(Queue\* q)

{

q->front = q->rear = -1;

}

void enqueue(Queue\* q, char\* item)

{

if (q->rear == MAX - 1)

{

printf("Queue Full!\n");

return;

}

if (q->front == -1)

q->front = 0;

q->items[++q->rear] = item;

}

char\* dequeue(Queue\* q)

{

if (q->front == -1 || q->front > q->rear)

return NULL;

return q->items[q->front++];

}

// Stack functions

void createStack(Stack\* s)

{

s->top = -1;

}

void push(Stack\* s, char\* item)

{

if (s->top == MAX - 1)

{

printf("Stack Overflow!\n");

return;

}

s->items[++s->top] = item;

}

char\* pop(Stack\* s)

{

if (s->top == -1)

return NULL;

return s->items[s->top--];

}

// Water usage log (Array)

void addLog(char\* logs[], int\* index, char\* usage)

{

if (\*index >= 5) {

for (int i = 0; i < 4; i++)

{

logs[i] = logs[i + 1];

}

logs[4] = usage;

}

else

{

logs[\*index] = usage;

(\*index)++;

}

}

// Singly linked list insert

Node\* insertSingly(Node\* head, char\* data)

{

Node\* newNode = (Node\*)malloc(sizeof(Node));

newNode->data = data;

newNode->next = head;

return newNode;

}

// Singly linked list remove

Node\* removeSingly(Node\* head, char\* data)

{

Node\* temp = head;

Node\* prev = NULL;

while (temp && strcmp(temp->data, data) != 0)

{

prev = temp;

temp = temp->next;

}

if (!temp) return head;

if (!prev)

head = temp->next;

else

prev->next = temp->next;

free(temp);

return head;

}

// Doubly linked list insert

dNode\* insertDoubly(dNode\* head, char\* data)

{

dNode\* newNode = (dNode\*)malloc(sizeof(dNode));

newNode->data = data;

newNode->prev = NULL;

newNode->next = head;

if (head) head->prev = newNode;

return newNode;

}

void displayForward(dNode\* head)

{

printf("Fixed Components (Forward): ");

while (head)

{

printf("%s ", head->data);

head = head->next;

}

printf("\n");

}

void displayBackward(dNode\* head)

{

if (!head) return;

while (head->next) head = head->next;

printf("Fixed Components (Backward): ");

while (head)

{

printf("%s ", head->data);

head = head->prev;

}

printf("\n");

}

// Circular linked list insert

cNode\* insertCircular(cNode\* head, char\* data)

{

cNode\* newNode = (cNode\*)malloc(sizeof(cNode));

newNode->data = data;

if (!head)

{

newNode->next = newNode;

return newNode;

}

cNode\* temp = head;

while (temp->next != head)

temp = temp->next;

temp->next = newNode;

newNode->next = head;

return head;

}

void traverseCircular(cNode\* head, int times)

{

if (!head) return;

cNode\* temp = head;

for (int i = 0; i < times; i++)

{

printf("Urgent Fix: %s\n", temp->data);

temp = temp->next;

}

}

int main()

{

Queue q;

Stack s;

createQueue(&q);

createStack(&s);

// a) Enqueue and push irrigation requests

char\* requests[] = {"Pump", "Pipe", "Valve", "Sensor", "Sprinkler", "Filter"};

for (int i = 0; i < 6; i++)

{

enqueue(&q, requests[i]);

}

while (q.front <= q.rear)

{

char\* item = dequeue(&q);

push(&s, item);

}

printf("Emergency Watering Order (LIFO):\n");

while (s.top >= 0)

{

printf("Watering: %s\n", pop(&s));

}

printf("\nLIFO suits emergencies because the last alert (e.g., 'Filter') is the most recent and may be most urgent.\n");

// b) Water usage log

char\* logs[5] = {NULL};

int logIndex = 0;

char\* usages[] = {"Use1", "Use2", "Use3", "Use4", "Use5", "Use6", "Use7"};

for (int i = 0; i < 7; i++)

{

addLog(logs, &logIndex, usages[i]);

}

printf("\nWater Usage Logs:\n");

for (int i = 0; i < 5; i++)

{

printf("Log[%d]: %s\n", i + 1, logs[i]);

}

printf("\nArchiving old logs helps analyze seasonal trends in water usage for better planning.\n");

// c) Maintenance Tracker

Node\* leakyList = NULL;

leakyList = insertSingly(leakyList, "Sprinkler");

leakyList = insertSingly(leakyList, "Pipe");

leakyList = removeSingly(leakyList, "Pipe");

dNode\* fixedList = NULL;

fixedList = insertDoubly(fixedList, "Pipe");

printf("\nComponent Maintenance:\n");

displayForward(fixedList);

displayBackward(fixedList);

printf("\nPipe cracked due to frost. It was sealed with resin and tested before being marked fixed.\n");

// d) High Priority Fixes

cNode\* priorityList = NULL;

priorityList = insertCircular(priorityList, "Pump");

priorityList = insertCircular(priorityList, "Valve");

printf("\nPriority Component Fixes:\n");

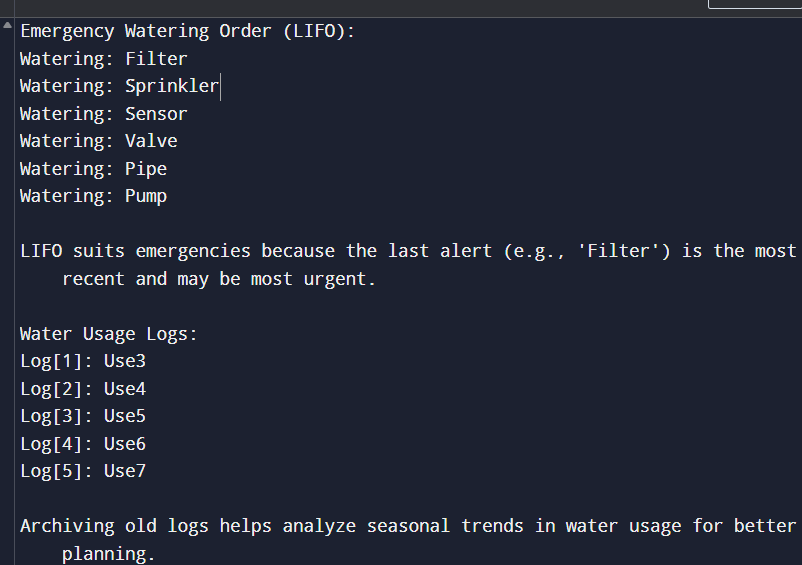
traverseCircular(priorityList, 4);

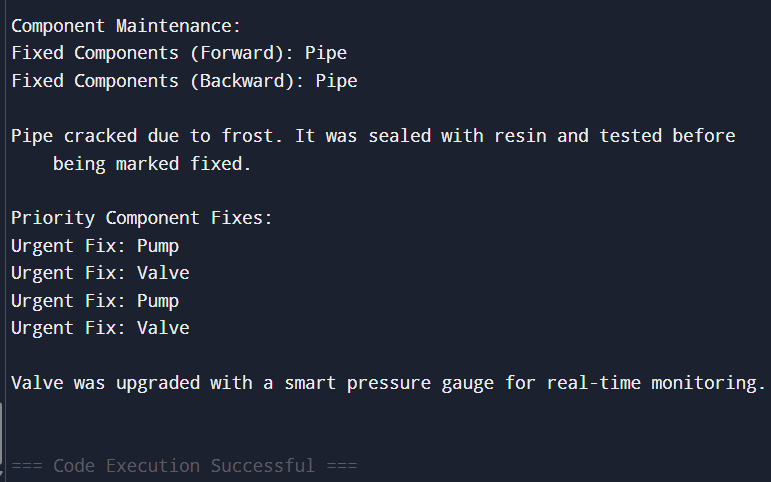
printf("\nValve was upgraded with a smart pressure gauge for real-time monitoring.\n");

return 0;

}

**OUTPUT**

****

****