Q. Implement linked list and its operations
Consider each node as structure representation of data for your domain. Perform all operations and implement different types of linked list

Singly.c

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

#define MAX_ITEM_NAME_LENGTH 50
#define MAX_PRICE_LENGTH 15
#define MAX_QUANTITY_LENGTH 5

// Node structure for food item information
struct MenuItemNode
{
    char itemName[MAX_ITEM_NAME_LENGTH];
    char price[MAX_PRICE_LENGTH];
    char quantity[MAX_QUANTITY_LENGTH];
    struct MenuItemNode *next;
};

typedef struct MenuItemNode MenuItemNode;

MenuItemNode *HEAD; // Head of the linked list

// Function to create a new menu item node
MenuItemNode *createMenuItemNode(const char *itemName, const char *price, const
char *quantity)
{
    MenuItemNode *newNode = (MenuItemNode *)malloc(sizeof(MenuItemNode));
    if (newNode == NULL)
    {
        printf("Memory allocation error.\n");
        exit(EXIT_FAILURE);
    }

    strncpy(newNode->itemName, itemName, MAX_ITEM_NAME_LENGTH - 1);
    newNode->itemName[MAX_ITEM_NAME_LENGTH - 1] = '\0'; // Ensure null-terminated
string

    strncpy(newNode->price, price, MAX_PRICE_LENGTH - 1);
    newNode->price[MAX_PRICE_LENGTH - 1] = '\0'; // Ensure null-terminated string
```

```
    strncpy(newNode->quantity, quantity, MAX_QUANTITY_LENGTH - 1);
    newNode->quantity[MAX_QUANTITY_LENGTH - 1] = '\0'; // Ensure null-terminated
string

    newNode->next = NULL;
    return newNode;
}

// Function to insert a new menu item at the beginning of the linked list
MenuItemNode *insertMenuItem(const char *itemName, const char *price, const char
*quantity)
{
    MenuItemNode *newNode = createMenuItemNode(itemName, price, quantity);
    newNode->next = HEAD;
    HEAD = newNode;
    return newNode;
}

// Function to delete a menu item by name
MenuItemNode *deleteMenuItem(const char *itemName)
{
    MenuItemNode *current = HEAD;
    MenuItemNode *prev = NULL;

    while (current != NULL)
    {
        if (strcmp(current->itemName, itemName) == 0)
        {
            if (prev == NULL)
            {
                // Deleting the first node
                HEAD = current->next;
            }
            else
            {
                prev->next = current->next;
            }

            free(current);
            return HEAD;
        }

        prev = current;
        current = current->next;
    }
```

```
    printf("Menu item '%s' not found in the menu.\n", itemName);
    return HEAD;
}

// Function to search for a menu item by name
void searchMenuItem(const char *itemName)
{
    MenuItemNode *current = HEAD;
    while (current != NULL)
    {
        if (strcmp(current->itemName, itemName) == 0)
        {
            printf("Menu Item Found: Name: %s, Price: %s, Quantity: %s\n",
current->itemName, current->price, current->quantity);
            return;
        }
        current = current->next;
    }
    printf("Menu item '%s' not found in the menu.\n", itemName);
}

// Function to display all menu items in the linked list
void displayMenu()
{
    MenuItemNode *current = HEAD;
    if (current == NULL)
    {
        printf("Menu is empty.\n");
        return;
    }

    printf("Menu Items:\n");
    while (current != NULL)
    {
        printf("Name: %s, Price: %s, Quantity: %s\n", current->itemName, current-
>price, current->quantity);
        current = current->next;
    }
}

// Function to free the memory occupied by the linked list
void freeMenu()
{
    MenuItemNode *current = HEAD;
```

```c
    while (current != NULL)
    {
        MenuItemNode *next = current->next;
        free(current);
        current = next;
    }
}


int main()
{
    MenuItemNode *menu = NULL;
    int choice;
    char itemName[MAX_ITEM_NAME_LENGTH];
    char price[MAX_PRICE_LENGTH];
    char quantity[MAX_QUANTITY_LENGTH];


    while (1)
    {
        printf("\nUniversity Canteen Management System (Linked List)\n");
        printf("1. Add Menu Item\n");
        printf("2. Delete Menu Item\n");
        printf("3. Search Menu Item\n");
        printf("4. Display Menu\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice)
        {
        case 1:
            printf("Enter menu item name to add: ");
            scanf("%s", itemName);
            printf("Enter the price: ");
            scanf("%s", price);
            printf("Enter the quantity: ");
            scanf("%s", quantity);
            menu = insertMenuItem(itemName, price, quantity);
            break;
        case 2:
            printf("Enter menu item name to delete: ");
            scanf("%s", itemName);
            menu = deleteMenuItem(itemName);
            break;
```

```
        case 3:
            printf("Enter menu item name to search: ");
            scanf("%s", itemName);
            searchMenuItem(itemName);
            break;
        case 4:
            displayMenu();
            break;
        case 5:
            freeMenu();
            printf("Exiting the program. Goodbye!\n");
            return 0;
        default:
            printf("Invalid choice. Please try again.\n");
        }
    }

    return 0;
}
```

Output-

a) Insertion

```
University Canteen Management System (Linked List)
1. Add Menu Item
2. Delete Menu Item
3. Search Menu Item
4. Display Menu
5. Exit
Enter your choice: 1
Enter menu item name to add: pizza
Enter the price: 456
Enter the quantity: 4

University Canteen Management System (Linked List)
1. Add Menu Item
2. Delete Menu Item
3. Search Menu Item
4. Display Menu
5. Exit
Enter your choice: 4
Menu Items:
Name: pizza, Price: 456, Quantity: 4
```

b) Deletion

```
University Canteen Management System (Linked List)
1. Add Menu Item
2. Delete Menu Item
3. Search Menu Item
4. Display Menu
5. Exit
Enter your choice: 2
Enter menu item name to delete: pizza

University Canteen Management System (Linked List)
1. Add Menu Item
2. Delete Menu Item
3. Search Menu Item
4. Display Menu
5. Exit
Enter your choice: 4
Menu is empty.
```

c) Search

```
University Canteen Management System (Linked List)
1. Add Menu Item
2. Delete Menu Item
3. Search Menu Item
4. Display Menu
5. Exit
Enter your choice: 1
Enter menu item name to add: pizza
Enter the price: 120
Enter the quantity: 4

University Canteen Management System (Linked List)
1. Add Menu Item
2. Delete Menu Item
3. Search Menu Item
4. Display Menu
5. Exit
Enter your choice: 1
Enter menu item name to add: burger
Enter the price: 80
Enter the quantity: 3

University Canteen Management System (Linked List)
1. Add Menu Item
2. Delete Menu Item
3. Search Menu Item
4. Display Menu
5. Exit
Enter your choice: 3
Enter menu item name to search: pizza
Menu Item Found: Name: pizza, Price: 120, Quantity: 4
```

d)Display

```
University Canteen Management System (Linked List)
1. Add Menu Item
2. Delete Menu Item
3. Search Menu Item
4. Display Menu
5. Exit
Enter your choice: 4
Menu Items:
Name: burger, Price: 80, Quantity: 3
Name: pizza, Price: 120, Quantity: 4
```

Doubly.c

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

#define MAX_ITEM_NAME_LENGTH 50
#define MAX_PRICE_LENGTH 15

// Node structure for food item information
struct MenuItemNode
{
    char itemName[MAX_ITEM_NAME_LENGTH];
    char price[MAX_PRICE_LENGTH];
    struct MenuItemNode *next;
    struct MenuItemNode *prev;
};

typedef struct MenuItemNode MenuItemNode;

MenuItemNode *HEAD;

// Function to create a new menu item node
MenuItemNode *createMenuItemNode(const char *itemName, const char *price)
{
    MenuItemNode *newNode = (MenuItemNode *)malloc(sizeof(MenuItemNode));
    if (newNode == NULL)
    {
        printf("Memory allocation error.\n");
        exit(EXIT_FAILURE);
    }

    strncpy(newNode->itemName, itemName, MAX_ITEM_NAME_LENGTH - 1);
    newNode->itemName[MAX_ITEM_NAME_LENGTH - 1] = '\0'; // Ensure null-terminated
string

    strncpy(newNode->price, price, MAX_PRICE_LENGTH - 1);
    newNode->price[MAX_PRICE_LENGTH - 1] = '\0'; // Ensure null-terminated string

    newNode->next = NULL;
    newNode->prev = NULL;
    return newNode;
}
```

```c
// Function to insert a new menu item at the end of the doubly linked list
MenuItemNode *insertMenuItem(const char *itemName, const char *price)
{
    MenuItemNode *newNode = createMenuItemNode(itemName, price);

    if (HEAD == NULL)
    {
        // If the list is empty, make the new node the HEAD
        HEAD = newNode;
    }
    else
    {
        MenuItemNode *current = HEAD;

        // Traverse to the last node
        while (current->next != NULL)
        {
            current = current->next;
        }

        // Insert the new node after the last node
        current->next = newNode;
        newNode->prev = current;
    }

    return HEAD;
}

// Function to delete a menu item by name
MenuItemNode *deleteMenuItem(const char *itemName)
{
    MenuItemNode *current = HEAD;
    while (current != NULL)
    {
        if (strcmp(current->itemName, itemName) == 0)
        {
            if (current->prev != NULL)
            {
                current->prev->next = current->next;
            }
            else
            {
                HEAD = current->next;
            }
```

```
            if (current->next != NULL)
            {
                current->next->prev = current->prev;
            }

            free(current);
            return HEAD;
        }
        current = current->next;
    }

    printf("Menu item '%s' not found in the menu.\n", itemName);
    return HEAD;
}

// Function to search for a menu item by name
void searchMenuItem(const char *itemName)
{
    MenuItemNode *current = HEAD;
    while (current != NULL)
    {
        if (strcmp(current->itemName, itemName) == 0)
        {
            printf("Menu Item Found: Name: %s, Price: %s\n", current->itemName,
current->price);
            return;
        }
        current = current->next;
    }
    printf("Menu item '%s' not found in the menu.\n", itemName);
}

// Function to display all menu items in the doubly linked list
void displayMenu()
{
    MenuItemNode *current = HEAD;
    if (current == NULL)
    {
        printf("Menu is empty.\n");
        return;
    }

    printf("Menu Items:\n");
    while (current != NULL)
    {
```

```
        printf("Name: %s, Price: %s\n", current->itemName, current->price);
        current = current->next;
    }
}

// Function to free the memory occupied by the doubly linked list
void freeMenu()
{
    MenuItemNode *current = HEAD;
    while (current != NULL)
    {
        MenuItemNode *next = current->next;
        free(current);
        current = next;
    }
}



int main()
{
    MenuItemNode *menu = NULL;
    int choice;
    char itemName[MAX_ITEM_NAME_LENGTH];
    char price[MAX_PRICE_LENGTH];

    loadMenu();

    while (1)
    {
        printf("\nUniversity Canteen Management System (Doubly Linked List)\n");
        printf("1. Add Menu Item\n");
        printf("2. Delete Menu Item\n");
        printf("3. Search Menu Item\n");
        printf("4. Display Menu\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice)
        {
        case 1:
            printf("Enter menu item name to add: ");
            scanf("%s", itemName);
            printf("Enter the price: ");
```

```
            scanf("%s", price);
            HEAD = insertMenuItem(itemName, price);
            break;
        case 2:
            printf("Enter menu item name to delete: ");
            scanf("%s", itemName);
            HEAD = deleteMenuItem(itemName);
            break;
        case 3:
            printf("Enter menu item name to search: ");
            scanf("%s", itemName);
            searchMenuItem(itemName);
            break;
        case 4:
            displayMenu();
            break;
        case 5:
            freeMenu();
            printf("Exiting the program. Goodbye!\n");
            return 0;
        default:
            printf("Invalid choice. Please try again.\n");
        }
    }

    return 0;
}
```

Output-

a) Insertion

```
University Canteen Management System (Doubly Linked List)
1. Add Menu Item
2. Delete Menu Item
3. Search Menu Item
4. Display Menu
5. Exit
Enter your choice: 1
Enter menu item name to add: pizza
Enter the price: 120

University Canteen Management System (Doubly Linked List)
1. Add Menu Item
2. Delete Menu Item
3. Search Menu Item
4. Display Menu
5. Exit
Enter your choice: 4
Menu Items:
Name: pizza, Price: 120
```

b) Deletion

```
University Canteen Management System (Doubly Linked List)
1. Add Menu Item
2. Delete Menu Item
3. Search Menu Item
4. Display Menu
5. Exit
Enter your choice: 2
Enter menu item name to delete: pizza

University Canteen Management System (Doubly Linked List)
1. Add Menu Item
2. Delete Menu Item
3. Search Menu Item
4. Display Menu
5. Exit
Enter your choice: 4
Menu is empty.
```

c) Search

```
University Canteen Management System (Doubly Linked List)
1. Add Menu Item
2. Delete Menu Item
3. Search Menu Item
4. Display Menu
5. Exit
Enter your choice: 1
Enter menu item name to add: pizza
Enter the price: 120

University Canteen Management System (Doubly Linked List)
1. Add Menu Item
2. Delete Menu Item
3. Search Menu Item
4. Display Menu
5. Exit
Enter your choice: 1
Enter menu item name to add: burger
Enter the price: 80

University Canteen Management System (Doubly Linked List)
1. Add Menu Item
2. Delete Menu Item
3. Search Menu Item
4. Display Menu
5. Exit
Enter your choice: 3
Enter menu item name to search: burger
Menu Item Found: Name: burger, Price: 80
```

4.) Display

```
University Canteen Management System (Doubly Linked List)
1. Add Menu Item
2. Delete Menu Item
3. Search Menu Item
4. Display Menu
5. Exit
Enter your choice: 4
Menu Items:
Name: pizza, Price: 120
Name: burger, Price: 80
```

Circular.c

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

#define MAX_ITEM_NAME_LENGTH 50
#define MAX_PRICE_LENGTH 15

// Node structure for menu item information
struct MenuItemNode
{
    char itemName[MAX_ITEM_NAME_LENGTH];
    char price[MAX_PRICE_LENGTH];
    struct MenuItemNode *next;
};

typedef struct MenuItemNode MenuItemNode;

MenuItemNode *HEAD;

// Function to create a new menu item node
MenuItemNode *createMenuItemNode(const char *itemName, const char *price)
{
    MenuItemNode *newNode = (MenuItemNode *)malloc(sizeof(MenuItemNode));
    if (newNode == NULL)
    {
        printf("Memory allocation error.\n");
        exit(EXIT_FAILURE);
    }

    strncpy(newNode->itemName, itemName, MAX_ITEM_NAME_LENGTH - 1);
    newNode->itemName[MAX_ITEM_NAME_LENGTH - 1] = '\0'; // Ensure null-terminated
string
```

```
    strncpy(newNode->price, price, MAX_PRICE_LENGTH - 1);
    newNode->price[MAX_PRICE_LENGTH - 1] = '\0'; // Ensure null-terminated string

    newNode->next = NULL;
    return newNode;
}

// Function to insert a new menu item into the circular linked list
MenuItemNode *insertMenuItem(const char *itemName, const char *price, int
position)
{
    MenuItemNode *newNode = createMenuItemNode(itemName, price);

    if (HEAD == NULL)
    {
        // If the list is empty, make the new node the HEAD and point it to
itself
        HEAD = newNode;
        newNode->next = HEAD;
    }
    else
    {
        // Traverse to the node at the desired position
        MenuItemNode *current = HEAD;
        int currentPosition = 1;

        while (currentPosition < position && current->next != HEAD)
        {
            current = current->next;
            currentPosition++;
        }

        // Insert the new node after the node at the desired position
        newNode->next = current->next;
        current->next = newNode;

        if (current == HEAD && position == 1)
        {
            // If inserting at the beginning and HEAD is updated, move HEAD to
the new node
            HEAD = newNode;
        }
    }
```

```
    return HEAD;
}

// Function to delete a menu item by name from the circular linked list
MenuItemNode *deleteMenuItem(const char *itemName)
{
    MenuItemNode *current = HEAD;
    MenuItemNode *prev = NULL;

    if (current == NULL)
    {
        printf("Menu is empty.\n");
        return HEAD;
    }

    do
    {
        if (strcmp(current->itemName, itemName) == 0)
        {
            if (current == HEAD)
            {
                // If deleting the first node, update HEAD
                HEAD = current->next;
            }

            if (prev != NULL)
            {
                // If not the first node, update the previous node's next pointer
                prev->next = current->next;
            }

            free(current);
            return HEAD;
        }

        prev = current;
        current = current->next;
    } while (current != HEAD);

    printf("Menu item '%s' not found in the menu.\n", itemName);
    return HEAD;
}

// Function to search for a menu item by name in the circular linked list
void searchMenuItem(const char *itemName)
```

```
{
    MenuItemNode *current = HEAD;

    if (current == NULL)
    {
        printf("Menu is empty.\n");
        return;
    }

    do
    {
        if (strcmp(current->itemName, itemName) == 0)
        {
            printf("Menu Item Found: Name: %s, Price: %s\n", current->itemName,
current->price);
            return;
        }
        current = current->next;
    } while (current != HEAD);

    printf("Menu item '%s' not found in the menu.\n", itemName);
}

// Function to display all menu items in the circular linked list
void displayMenu()
{
    MenuItemNode *current = HEAD;

    if (current == NULL)
    {
        printf("Menu is empty.\n");
        return;
    }

    printf("Menu Items:\n");
    do
    {
        printf("Name: %s, Price: %s\n", current->itemName, current->price);
        current = current->next;
    } while (current != HEAD);
}

// Function to free the memory occupied by the circular linked list
void freeMenu()
{
```

```
    MenuItemNode *current = HEAD;

    if (current == NULL)
    {
        return;
    }

    do
    {
        MenuItemNode *next = current->next;
        free(current);
        current = next;
    } while (current != HEAD);
}



int main()
{
    MenuItemNode *menu = NULL;
    int choice;
    char itemName[MAX_ITEM_NAME_LENGTH];
    char price[MAX_PRICE_LENGTH];
    int pos = 0; // By default, insert at the start

    loadMenu(); // Loading menu items

    while (1)
    {
        printf("\nUniversity Canteen Management System (Circular Linked
List)\n");
        printf("1. Add Menu Item\n");
        printf("2. Delete Menu Item\n");
        printf("3. Search Menu Item\n");
        printf("4. Display Menu\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice)
        {
        case 1:
            printf("Enter menu item name to add: ");
            scanf("%s", itemName);
            printf("Enter the price: ");
```

```
            scanf("%s", price);
            printf("Enter the position: ");
            scanf("%d", &pos);
            HEAD = insertMenuItem(itemName, price, pos);
            break;
        case 2:
            printf("Enter menu item name to delete: ");
            scanf("%s", itemName);
            HEAD = deleteMenuItem(itemName);
            break;
        case 3:
            printf("Enter menu item name to search: ");
            scanf("%s", itemName);
            searchMenuItem(itemName);
            break;
        case 4:
            displayMenu();
            break;
        case 5:
            freeMenu();
            printf("Exiting the program. Goodbye!\n");
            return 0;
        default:
            printf("Invalid choice. Please try again.\n");
        }
    }

    return 0;
}
```

Output-

a) Insertion

```
University Canteen Management System (Circular Linked List)
1. Add Menu Item
2. Delete Menu Item
3. Search Menu Item
4. Display Menu
5. Exit
Enter your choice: 1
Enter menu item name to add: pizza
Enter the price: 120
Enter the position: 0

University Canteen Management System (Circular Linked List)
1. Add Menu Item
2. Delete Menu Item
3. Search Menu Item
4. Display Menu
5. Exit
Enter your choice: 4
Menu Items:
Name: pizza, Price: 120
```

b) Delete

```
University Canteen Management System (Circular Linked List)
1. Add Menu Item
2. Delete Menu Item
3. Search Menu Item
4. Display Menu
5. Exit
Enter your choice: 1
Enter menu item name to add: pizza
Enter the price: 120
Enter the position: 0

University Canteen Management System (Circular Linked List)
1. Add Menu Item
2. Delete Menu Item
3. Search Menu Item
4. Display Menu
5. Exit
Enter your choice: 2
Enter menu item name to delete: pizza

University Canteen Management System (Circular Linked List)
1. Add Menu Item
2. Delete Menu Item
3. Search Menu Item
4. Display Menu
5. Exit
Enter your choice: 4
Menu Items:
Name: ⌂L º, Price: 120
```

c) Search

```
University Canteen Management System (Circular Linked List)
1. Add Menu Item
2. Delete Menu Item
3. Search Menu Item
4. Display Menu
5. Exit
Enter your choice: 1
Enter menu item name to add: pizza
Enter the price: 120
Enter the position: 0

University Canteen Management System (Circular Linked List)
1. Add Menu Item
2. Delete Menu Item
3. Search Menu Item
4. Display Menu
5. Exit
Enter your choice: 1
Enter menu item name to add: burger
Enter the price: 80
Enter the position: 5

University Canteen Management System (Circular Linked List)
1. Add Menu Item
2. Delete Menu Item
3. Search Menu Item
4. Display Menu
5. Exit
Enter your choice: 3
Enter menu item name to search: pizza
Menu Item Found: Name: pizza, Price: 120
```

d) Display

```
University Canteen Management System (Circular Linked List)
1. Add Menu Item
2. Delete Menu Item
3. Search Menu Item
4. Display Menu
5. Exit
Enter your choice: 3
Enter menu item name to search: pizza
Menu Item Found: Name: pizza, Price: 120

University Canteen Management System (Circular Linked List)
1. Add Menu Item
2. Delete Menu Item
3. Search Menu Item
4. Display Menu
5. Exit
Enter your choice: 4
Menu Items:
Name: pizza, Price: 120
Name: burger, Price: 80
```