Experiment No.7:

Stack or Queue using Linked List (Dynamic Implementation)
Design a service window system where customers arrive and are served in order (FIFO),or a browser history system where the last visited page is accessed first (LIFO).Use a linked list to implement a dynamic stack (push, pop, display) or queue (add,delete, display) based on the given use case.

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

// Define Node structure
struct Node {
   char data[100];
   struct Node* next;
};

// Stack pointers
struct Node* top = NULL;

// Queue pointers
struct Node* front = NULL;
struct Node* rear = NULL;

// Stack Functions - Browser History (LIFO)
void push(char value[]) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   strcpy(newNode->data, value);
   newNode->next = top;
   top = newNode;
   printf("Page visited: %s\n", value);
}

void pop() {
   if (top == NULL) {
      printf("No page in history.\n");
      return;
   }
   struct Node* temp = top;
   printf("Back to page: %s\n", temp->data);
   top = top->next;
   free(temp);
}

void displayStack() {
   if (top == NULL) {
      printf("History is empty.\n");
      return;
   }
```

```
    struct Node* temp = top;
    printf("Browser History:\n");
    while (temp != NULL) {
        printf("%s\n", temp->data);
        temp = temp->next;
    }
}

// Queue Functions - Service Window (FIFO)
void enqueue(char value[]) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    strcpy(newNode->data, value);
    newNode->next = NULL;
    if (rear == NULL) {
        front = rear = newNode;
    } else {
        rear->next = newNode;
        rear = newNode;
    }
    printf("Customer arrived: %s\n", value);
}

void dequeue() {
    if (front == NULL) {
        printf("No customer in queue.\n");
        return;
    }
    struct Node* temp = front;
    printf("Customer served: %s\n", temp->data);
    front = front->next;
    if (front == NULL) rear = NULL;
    free(temp);
}

void displayQueue() {
    if (front == NULL) {
        printf("Queue is empty.\n");
        return;
    }
    struct Node* temp = front;
    printf("Waiting Customers:\n");
    while (temp != NULL) {
        printf("%s\n", temp->data);
        temp = temp->next;
    }
}

int main() {
    int choice, option;
    char name[100];
```

```
printf("Choose System:\n");
printf("1. Service Window (Queue)\n");
printf("2. Browser History (Stack)\n");
printf("Enter your choice: ");
scanf("%d", &option);
getchar(); // clear newline

if (option == 1) {
   // Queue Menu
   do {
      printf("\n--- Service Window (Queue) ---\n");
      printf("1. Add Customer\n");
      printf("2. Serve Customer\n");
      printf("3. Show Queue\n");
      printf("4. Exit\n");
      printf("Enter choice: ");
      scanf("%d", &choice);
      getchar();

      switch (choice) {
         case 1:
            printf("Enter customer name: ");
            fgets(name, sizeof(name), stdin);
            name[strcspn(name, "\n")] = 0;
            enqueue(name);
            break;
         case 2: dequeue(); break;
         case 3: displayQueue(); break;
         case 4: printf("Exiting queue system.\n"); break;
         default: printf("Invalid choice.\n");
      }
   } while (choice != 4);
}
else if (option == 2) {
   // Stack Menu
   do {
      printf("\n--- Browser History (Stack) ---\n");
      printf("1. Visit Page\n");
      printf("2. Go Back\n");
      printf("3. Show History\n");
      printf("4. Exit\n");
      printf("Enter choice: ");
      scanf("%d", &choice);
      getchar();

      switch (choice) {
         case 1:
            printf("Enter page URL: ");
            fgets(name, sizeof(name), stdin);
```

```
                name[strcspn(name, "\n")] = 0;
                push(name);
                break;
            case 2: pop(); break;
            case 3: displayStack(); break;
            case 4: printf("Exiting browser history.\n"); break;
            default: printf("Invalid choice.\n");
        }
    } while (choice != 4);
  }
  else {
    printf("Invalid system choice.\n");
  }

  return 0;
}
```