ASSIGNMENT-9

1. Write a menu driven program to create a Stack using Linked List

SOLUTION:

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

struct Node {
    int data;
    struct Node* next;
};

struct Stack {
    struct Node* top;
    int size;
};

struct Node* newNode(int data) {
    struct Node* node = (struct Node*)malloc(sizeof(struct Node));
    node->data = data;
    node->next = NULL;
    return node;
}

struct Stack* createStack() {
    struct Stack* stack = (struct Stack*)malloc(sizeof(struct Stack));
    stack->top = NULL;
    stack->size = 0;
    return stack;
}

int isEmpty(struct Stack* stack) {
    return (stack->top == NULL);
}

void push(struct Stack* stack, int data) {
    struct Node* node = newNode(data);
    node->next = stack->top;
    stack->top = node;
    stack->size++;
    printf("Pushed %d onto the stack.\n", data);
}

int pop(struct Stack* stack) {
    if (isEmpty(stack)) {
```

```
        printf("Stack Underflow!\n");
        exit(1);
    }
    struct Node* node = stack->top;
    int data = node->data;
    stack->top = node->next;
    free(node);
    stack->size--;
    printf("Popped %d from the stack.\n", data);
    return data;
}

void display(struct Stack* stack) {
    if (isEmpty(stack)) {
        printf("Stack is empty.\n");
        return;
    }
    struct Node* current = stack->top;
    printf("Stack:\n");
    while (current != NULL) {
        printf("%d\n", current->data);
        current = current->next;
    }
    printf("\n");
}

int main() {
    struct Stack* stack = createStack();
    int choice, data;

    while (1) {
        printf("\nStack Menu:\n");
        printf("1. Push\n");
        printf("2. Pop\n");
        printf("3. Display\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter the value to push onto the stack: ");
                scanf("%d", &data);
                push(stack, data);
                break;

            case 2:
                if (isEmpty(stack)) {
                    printf("Stack Underflow!\n");
```

```
            } else {
                pop(stack);
            }
            break;

        case 3:
            display(stack);
            break;

        case 4:
            printf("Exiting the program.\n");
            exit(0);

        default:
            printf("Invalid choice. Please try again.\n");
      }
   }
   return 0;
}
```

OUTPUT:

```
Stack Menu:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 1
Enter the value to push onto the stack: 10
Pushed 10 onto the stack.

Stack Menu:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 1
Enter the value to push onto the stack: 20
Pushed 20 onto the stack.

Stack Menu:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 1
Enter the value to push onto the stack: 30
Pushed 30 onto the stack.
```

```
Stack Menu:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 3
Stack:
30
20
10


Stack Menu:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 2
Popped 30 from the stack.

Stack Menu:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 3
Stack:
20
10
```

2. Write a menu driven program to create a Queue using Linked List

SOLUTION:

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

struct Node {
    int data;
    struct Node* next;
};

struct Queue {
    struct Node* front;
    struct Node* rear;
    int size;
    int max_size;
```

```
};

struct Node* newNode(int data) {
    struct Node* node = (struct Node*)malloc(sizeof(struct Node));
    node->data = data;
    node->next = NULL;
    return node;
}

struct Queue* createQueue(int max_size) {
    struct Queue* queue = (struct Queue*)malloc(sizeof(struct Queue));
    queue->front = queue->rear = NULL;
    queue->size = 0;
    queue->max_size = max_size;
    return queue;
}

int isEmpty(struct Queue* queue) {
    return (queue->front == NULL);
}

int isFull(struct Queue* queue) {
    return (queue->size >= queue->max_size);
}

void enqueue(struct Queue* queue, int data) {
    if (isFull(queue)) {
        printf("Queue Overflow!\n");
        return;
    }
    struct Node* node = newNode(data);
    if (isEmpty(queue)) {
        queue->front = queue->rear = node;
    } else {
        queue->rear->next = node;
        queue->rear = node;
    }
    queue->size++;
    printf("Enqueued %d into the queue.\n", data);
}

int dequeue(struct Queue* queue) {
    if (isEmpty(queue)) {
        printf("Queue Underflow!\n");
        exit(1);
    }
    struct Node* node = queue->front;
    int data = node->data;
```

```
    queue->front = node->next;
    free(node);
    queue->size--;
    return data;
}

void display(struct Queue* queue) {
    if (isEmpty(queue)) {
        printf("Queue is empty.\n");
        return;
    }
    struct Node* current = queue->front;
    printf("Queue: ");
    while (current != NULL) {
        printf("%d ", current->data);
        current = current->next;
    }
    printf("\n");
}

int main() {
    int max_size, choice, data;
    printf("Enter the max. queue size: ");
    scanf("%d", &max_size);
    if (max_size <= 0) {
        printf("Invalid queue size!\n");
        return 1;
    }
    struct Queue* queue = createQueue(max_size);
    while (1) {
        printf("\nQueue Menu:\n");
        printf("1. Insert (Enqueue)\n");
        printf("2. Delete (Dequeue)\n");
        printf("3. Display\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d",&choice);
        switch (choice) {
            case 1:
                if (!isFull(queue)) {
                    printf("Enter the value to enqueue: ");
                    scanf("%d", &data);
                    enqueue(queue, data);
                } else {
                    printf("Queue Overflow!\n");
                }
                break;
```

```
            case 2:
                if (isEmpty(queue)) {
                    printf("Queue Underflow!\n");
                } else {
                    printf("Dequeued %d from the queue.\n", dequeue(queue));
                }
                break;

            case 3:
                display(queue);
                break;

            case 4:
                printf("Exiting the program.\n");
                exit(0);

            default:
                printf("Invalid choice!\n");
        }
    }
    return 0;
}
```

OUTPUT:

```
Enter the max. queue size: 3

Queue Menu:
1. Insert (Enqueue)
2. Delete (Dequeue)
3. Display
4. Exit
Enter your choice: 1
Enter the value to enqueue: 10
Enqueued 10 into the queue.

Queue Menu:
1. Insert (Enqueue)
2. Delete (Dequeue)
3. Display
4. Exit
Enter your choice: 1
Enter the value to enqueue: 20
Enqueued 20 into the queue.

Queue Menu:
1. Insert (Enqueue)
2. Delete (Dequeue)
```

3. Display
4. Exit
Enter your choice: 1
Enter the value to enqueue: 30
Enqueued 30 into the queue.

Queue Menu:
1. Insert (Enqueue)
2. Delete (Dequeue)
3. Display
4. Exit
Enter your choice: 3
Queue: 10 20 30

Queue Menu:
1. Insert (Enqueue)
2. Delete (Dequeue)
3. Display
4. Exit
Enter your choice: 1
Queue Overflow!