Data Structures Week 6:

Name: Aaryan Prakash

USN: 1BM23SC006

Class: 3A

Q) WAP to Implement Single Link List to simulate Stack & Queue Operations.

Code:

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

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

typedef struct Node node;

//Stack
node *top=NULL;
void push();
void pop();
void displayStack();

void push(){
   node *new1=(node*)malloc(sizeof(node));
   if(new1==NULL){
      printf("\nStack Overflow.\n");
      return;
   }
```

```
    printf("\nEnter Value to Push: ");
    scanf("%d", &new1->data);
    new1->link=top;
    top=new1;
}

void pop(){
    if(top==NULL){
        printf("\nStack Underflow.\n");
        return;
    }

    node *temp=top;
    printf("\nPopped Element: %d\n", temp->data);
    top=top->link;
    free(temp);
}

void displayStack(){
    if(top==NULL){
        printf("\nThe Stack is Empty.\n");
        return;
    }

    printf("\nElements in the Stack: ");
    node *temp=top;
    while(temp!=NULL){
        printf("%d ", temp->data);
        temp=temp->link;
    }
    printf("\n");
```

```
}


//Queue
node *front=NULL, *rear=NULL;

void insert();
void del();
void displayQueue();

void insert(){
   node *new1=(node*)malloc(sizeof(node));
   if(new1==NULL){
      printf("\nQueue Full.\n");
      return;
   }

   printf("\nEnter Value to Insert: ");
   scanf("%d", &new1->data);
   new1->link=NULL;

   if(rear==NULL){
      front=rear=new1;
      return;
   }
   rear->link=new1;
   rear=new1;
}

void del(){
   if(front==NULL){
```

```
        printf("\nQueue Empty.\n");
        return;
    }

    node *temp=front;
    printf("\nDeleted Element: %d\n", temp->data);
    front=front->link;

    if(front==NULL){
        rear=NULL;
    }
    free(temp);
}

void displayQueue(){
    if(front==NULL){
        printf("\nThe Queue is Empty.\n");
        return;
    }

    printf("\nElements in the Queue: ");
    node *temp=front;
    while(temp!=NULL){
        printf("%d ", temp->data);
        temp=temp->link;
    }
    printf("\n");
}


// Main
```

```
void main(){
   int ch;


   while(1){
      printf("\n1. Push (Stack) \n2. Pop (Stack) \n3. Display (Stack)");
      printf("\n4. Insert (Queue) \n5. Delete (Queue) \n6. Display (Queue) \n7.
Exit");
      printf("\nEnter Your Choice: ");
      scanf("%d", &ch);


      switch(ch){
         case 1:
            push();
            break;
         case 2:
            pop();
            break;
         case 3:
            displayStack();
            break;
         case 4:
            insert();
            break;
         case 5:
            del();
            break;
         case 6:
            displayQueue();
            break;
         case 7:
            exit(0);
```

```
        default:
            printf("\nEnter Your Choice: \n");
      }
    }
}
```

Output:

```
1. Push (Stack)
2. Pop (Stack)
3. Display (Stack)
4. Insert (Queue)
5. Delete (Queue)
6. Display (Queue)
7. Exit
Enter Your Choice: 1

Enter Value to Push: 10

1. Push (Stack)
2. Pop (Stack)
3. Display (Stack)
4. Insert (Queue)
5. Delete (Queue)
6. Display (Queue)
7. Exit
Enter Your Choice: 1

Enter Value to Push: 20

1. Push (Stack)
2. Pop (Stack)
3. Display (Stack)
4. Insert (Queue)
5. Delete (Queue)
6. Display (Queue)
7. Exit
Enter Your Choice: 1

Enter Value to Push: 30

1. Push (Stack)
2. Pop (Stack)
3. Display (Stack)
4. Insert (Queue)
5. Delete (Queue)
6. Display (Queue)
7. Exit
Enter Your Choice: 1

Enter Value to Push: 40
```

```
1. Push (Stack)
2. Pop (Stack)
3. Display (Stack)
4. Insert (Queue)
5. Delete (Queue)
6. Display (Queue)
7. Exit
Enter Your Choice: 3

Elements in the Stack: 40 30 20 10

1. Push (Stack)
2. Pop (Stack)
3. Display (Stack)
4. Insert (Queue)
5. Delete (Queue)
6. Display (Queue)
7. Exit
Enter Your Choice: 2

Popped Element: 40

1. Push (Stack)
2. Pop (Stack)
3. Display (Stack)
4. Insert (Queue)
5. Delete (Queue)
6. Display (Queue)
7. Exit
Enter Your Choice: 2

Popped Element: 30

1. Push (Stack)
2. Pop (Stack)
3. Display (Stack)
4. Insert (Queue)
5. Delete (Queue)
6. Display (Queue)
7. Exit
Enter Your Choice: 3

Elements in the Stack: 20 10
```

```
1. Push (Stack)
2. Pop (Stack)
3. Display (Stack)
4. Insert (Queue)
5. Delete (Queue)
6. Display (Queue)
7. Exit
Enter Your Choice: 4

Enter Value to Insert: 10

1. Push (Stack)
2. Pop (Stack)
3. Display (Stack)
4. Insert (Queue)
5. Delete (Queue)
6. Display (Queue)
7. Exit
Enter Your Choice: 4

Enter Value to Insert: 20

1. Push (Stack)
2. Pop (Stack)
3. Display (Stack)
4. Insert (Queue)
5. Delete (Queue)
6. Display (Queue)
7. Exit
Enter Your Choice: 4

Enter Value to Insert: 30

1. Push (Stack)
2. Pop (Stack)
3. Display (Stack)
4. Insert (Queue)
5. Delete (Queue)
6. Display (Queue)
7. Exit
Enter Your Choice: 4

Enter Value to Insert: 40
```

```
1. Push (Stack)
2. Pop (Stack)
3. Display (Stack)
4. Insert (Queue)
5. Delete (Queue)
6. Display (Queue)
7. Exit
Enter Your Choice: 6

Elements in the Queue: 10 20 30 40

1. Push (Stack)
2. Pop (Stack)
3. Display (Stack)
4. Insert (Queue)
5. Delete (Queue)
6. Display (Queue)
7. Exit
Enter Your Choice: 5

Deleted Element: 10

1. Push (Stack)
2. Pop (Stack)
3. Display (Stack)
4. Insert (Queue)
5. Delete (Queue)
6. Display (Queue)
7. Exit
Enter Your Choice: 5

Deleted Element: 20

1. Push (Stack)
2. Pop (Stack)
3. Display (Stack)
4. Insert (Queue)
5. Delete (Queue)
6. Display (Queue)
7. Exit
Enter Your Choice: 5

Deleted Element: 30
```

```
1. Push (Stack)
2. Pop (Stack)
3. Display (Stack)
4. Insert (Queue)
5. Delete (Queue)
6. Display (Queue)
7. Exit
Enter Your Choice: 6

Elements in the Queue: 40

1. Push (Stack)
2. Pop (Stack)
3. Display (Stack)
4. Insert (Queue)
5. Delete (Queue)
6. Display (Queue)
7. Exit
Enter Your Choice: 7
```

Q) WAP to Implement Single Link List with following operations:

- **Sort the linked list**
- **Reverse the linked list**
- **Concatenation of two linked lists.**

Code:

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

struct Node {
   int data;
   struct Node *link;
};
typedef struct Node node;

node *start = NULL, *temp, *new1, *curr;
int ch;
```

```
char c;

void createList();
void sort();
void reverse();
void display();
void concatenate();

void createList() {
   do {
      new1 = (node*)malloc(sizeof(node));
      printf("Enter Value: ");
      scanf("%d", &new1->data);
      new1->link = NULL;

      if (start == NULL) {
         start = new1;
         curr = new1;
      } else {
         curr->link = new1;
         curr = new1;
      }
      printf("Do you want to add another element (Y/N): ");
      scanf(" %c", &c);
   } while (c == 'y' || c == 'Y');
}

void sort() {
   if (start == NULL) {
      printf("The Linked List is Empty.\n");
      return;
```

```
    }

    node *i, *j;
    int tempData;
    for (i = start; i != NULL; i = i->link) {
        for (j = i->link; j != NULL; j = j->link) {
            if (i->data > j->data) {
                tempData = i->data;
                i->data = j->data;
                j->data = tempData;
            }
        }
    }
    printf("Linked List is Sorted.\n");
}

void reverse() {
    node *a = start, *b = NULL;
    while (a != NULL) {
        temp = a->link;
        a->link = b;
        b = a;
        a = temp;
    }
    start = b;
    printf("Linked List is Reversed.\n");
}

void display() {
    if (start == NULL) {
        printf("Linked list is Empty\n");
```

```
        return;
    }

    temp = start;
    printf("Elements in Linked List:\n");
    while (temp != NULL) {
        printf("%d\t", temp->data);
        temp = temp->link;
    }
    printf("\n");
}

void concatenate() {
    node *start2 = NULL, *curr2 = NULL;

    printf("Enter the second linked list:\n");
    createList();

    do {
        new1 = (node*)malloc(sizeof(node));
        printf("Enter value for second list: ");
        scanf("%d", &new1->data);
        new1->link = NULL;

        if (start2 == NULL) {
            start2 = new1;
            curr2 = new1;
        } else {
            curr2->link = new1;
            curr2 = new1;
        }
```

```
        printf("Do you want to add another element (Y/N): ");
        scanf(" %c", &c);
    } while (c == 'y' || c == 'Y');

    if (start == NULL) {
        start = start2;
    } else {
        temp = start;
        while (temp->link != NULL) {
            temp = temp->link;
        }
        temp->link = start2;
    }
    start2 = NULL;
    printf("Lists concatenated successfully.\n");
}

int main() {
    while (1) {
        printf("\n1. Create 1st Linked List\n2. Sort Linked List\n3. Reverse Linked
List\n4. Concatenate Linked Lists\n5. Display Linked List\n6. Exit\n");
        printf("Enter Your Choice: ");
        scanf("%d", &ch);
        switch (ch) {
            case 1:
                createList();
                break;
            case 2:
                sort();
                break;
            case 3:
```

```
            reverse();
            break;
        case 4:
            concatenate();
            break;
        case 5:
            display();
            break;
        case 6:
            exit(0);
            break;
        default:
            printf("Invalid choice. Please try again.\n");
            break;
      }
   }
}
```

Output:

```
1. Create 1st Linked List
2. Sort Linked List
3. Reverse Linked List
4. Concatenate Linked Lists
5. Display Linked List
6. Exit
Enter Your Choice: 1
Enter Value: 10
Do you want to add another element (Y/N): y
Enter Value: 80
Do you want to add another element (Y/N): y
Enter Value: 60
Do you want to add another element (Y/N): y
Enter Value: 20
Do you want to add another element (Y/N): y
Enter Value: 70
Do you want to add another element (Y/N): y
Enter Value: 30
Do you want to add another element (Y/N): n

1. Create 1st Linked List
2. Sort Linked List
3. Reverse Linked List
4. Concatenate Linked Lists
5. Display Linked List
6. Exit
Enter Your Choice: 5
Elements in Linked List:
10      80      60      20      70      30

1. Create 1st Linked List
2. Sort Linked List
3. Reverse Linked List
4. Concatenate Linked Lists
5. Display Linked List
6. Exit
Enter Your Choice: 3
Linked List is Reversed.

1. Create 1st Linked List
2. Sort Linked List
3. Reverse Linked List
4. Concatenate Linked Lists
5. Display Linked List
6. Exit
Enter Your Choice: 5
Elements in Linked List:
30      70      20      60      80      10
```

```
1. Create 1st Linked List
2. Sort Linked List
3. Reverse Linked List
4. Concatenate Linked Lists
5. Display Linked List
6. Exit
Enter Your Choice: 2
Linked List is Sorted.

1. Create 1st Linked List
2. Sort Linked List
3. Reverse Linked List
4. Concatenate Linked Lists
5. Display Linked List
6. Exit
Enter Your Choice: 5
Elements in Linked List:
10      20      30      60      70      80

1. Create 1st Linked List
2. Sort Linked List
3. Reverse Linked List
4. Concatenate Linked Lists
5. Display Linked List
6. Exit
Enter Your Choice: 4
Enter the second linked list:
Enter Value: 10
Do you want to add another element (Y/N): y
Enter Value: 70
Do you want to add another element (Y/N): y
Enter Value: 80
Do you want to add another element (Y/N): y
Enter Value: 60
Do you want to add another element (Y/N): y
Enter Value: 30
Do you want to add another element (Y/N): n
Enter value for second list: 10
Do you want to add another element (Y/N): y
Enter value for second list: 50
Do you want to add another element (Y/N): y
Enter value for second list: 60
Do you want to add another element (Y/N): y
Enter value for second list: 40
Do you want to add another element (Y/N): n
Lists concatenated successfully.
```

```
1. Create 1st Linked List
2. Sort Linked List
3. Reverse Linked List
4. Concatenate Linked Lists
5. Display Linked List
6. Exit
Enter Your Choice: 5
Elements in Linked List:
10      10      70      80      60      30      10      50      60      40

1. Create 1st Linked List
2. Sort Linked List
3. Reverse Linked List
4. Concatenate Linked Lists
5. Display Linked List
6. Exit
Enter Your Choice: 2
Linked List is Sorted.

1. Create 1st Linked List
2. Sort Linked List
3. Reverse Linked List
4. Concatenate Linked Lists
5. Display Linked List
6. Exit
Enter Your Choice: 5
Elements in Linked List:
10      10      10      30      40      50      60      60      70      80

1. Create 1st Linked List
2. Sort Linked List
3. Reverse Linked List
4. Concatenate Linked Lists
5. Display Linked List
6. Exit
Enter Your Choice: 3
Linked List is Reversed.

1. Create 1st Linked List
2. Sort Linked List
3. Reverse Linked List
4. Concatenate Linked Lists
5. Display Linked List
6. Exit
Enter Your Choice: 5
Elements in Linked List:
80      70      60      60      50      40      30      10      10      10
```