Lab - 5

1) Singly linked list - sort, reverse, concatenation

ans)

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

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

void insert (struct node** head, int data)
{
    struct node* newnode = (struct node*) malloc
                           (sizeof (struct node));
    newnode -> data = data;
    newnode -> next = * head;
    * head = newnode
}

void print ( struct node* head)
{
    while (head != NULL)
    {
        printf ("%d\t", head -> data);
        head = head -> next;
    }
}

void sort ( struct node** head)
{
```

```
struct node * current, * nextnode;
int temp;
current = * head;
while (current != NULL)
{
    nextnode = current -> next;
    while (nextnode != NULL)
    {
        if (current -> data > nextnode -> data)
        {
            temp = current -> data;
            current -> data = nextnode -> data;
            nextnode -> data = temp;
        }
        nextnode = nextnode -> next;
    }
    current = current -> next;
}
}

void reverse ( struct node ** head)
{
    struct node * prev, * current, * nextnode;
    prev = NULL;
    current = * head;
    while (current != NULL)
    {
        nextnode = current -> next;
        current -> next = prev;
        prev = current;
        current = nextnode;
    }
    * head = prev;
}
```

```
void concatenate (struct node ** list1, struct node ** list2)
{
    if (*list1 == NULL)
    {
        *list1 = *list2;
        return;
    }
    struct node * temp = *list1;
    while (temp -> next != NULL)
    {
        temp = temp -> next;
    }
    temp -> next = list2;
}

void main()
{
    struct node * list1 = NULL;
    struct node * list2 = NULL;
    int choice;
    int data;
    while (1)
    {
        printf (" Menu \n 1. Insert into list 1 \t
        2. Insert into list 2 \t 3. Sort list 1 \t
        4. concatenate lists \t 5. Reverse list 1 \t
        6. Display \t 7. Exit \n");
        printf (" Enter your choice: ");
        scanf ("%d", &choice);
        switch (choice)
        {
            case 1:
                printf (" Enter the data to
                insert into list 1 :");
```

```
            scanf("%d", &data);
            insert(&list1, data);
            break;
        case 2:
            printf("Enter data to insert in list 2: ");
            scanf("%d", &data);
            insert(&list2, data);
            break;
        case 3:
            sort(&list1);
            printf("list 1 sorted");
            break;
        case 4:
            concatenate(&list1, &list2);
            printf("lists concatenated");
            break;
        case 5:
            reverse(&list1);
            printf("list 1 reversed");
            break;
        case 6:
            printf("list 1: ");
            print(list1);
            break;
        case 7:
            exit(0);
            break;
        default:
            printf("Invalid input");
    }
}
}
```

OUTPUT:

Menu

1. Insert into list 1 2. Insert into list 2
3. Sort list 4. Concatenate lists 5. Reverse list 6. Display 7. Exit

Enter your choice: 1

Enter the data to insert into list 1: 24

Menu

1. Insert into list 1 2. Insert into list 2 3. Sort list
4. Concatenate lists 5. Reverse list 6. Display 7. Exit

Enter your choice: 1

Enter the data to insert into list 1: 2

Menu

1. Insert into list 1 2. Insert into list 2 3. Sort list
4. Concatenate lists 5. Reverse list 6. Display 7. Exit

Enter your choice: 1

Enter the data to insert into list 1: 3

Menu

1. Insert into list 1 2. Insert into list 2 3. Sort list
4. Concatenate list 5. Reverse list 6. Display 7. Exit

Enter your choice: 3

List 1 sorted

Menu

1. Insert into list 1 2. Insert into list 2 3. Sort list
4. Concatenate lists 5. Reverse list 6. Display 7. Exit

Enter your choice: 6

List 1: 2 3 4

Menu

1. Insert into list 1 2. Insert into list 2 3. Sort list
4. Concatenate list 5. Reverse list 6. Display 7. Exit

Enter your choice: 7

```
Menu

1. Insert into List 1
2. Insert into List 2
3. Sort List 1
4. Reverse List 2
5. Concatenate Lists
6. Print Lists
0. Exit
Enter your choice: 1
Enter data to insert into List 1: 3
Menu

1. Insert into List 1
2. Insert into List 2
3. Sort List 1
4. Reverse List 2
5. Concatenate Lists
6. Print Lists
0. Exit
Enter your choice: 1
Enter data to insert into List 1: 2
Menu

1. Insert into List 1
2. Insert into List 2
3. Sort List 1
4. Reverse List 2
5. Concatenate Lists
6. Print Lists
0. Exit
Enter your choice: 1
Enter data to insert into List 1: 9
```

```
Menu

1. Insert into List 1
2. Insert into List 2
3. Sort List 1
4. Reverse List 2
5. Concatenate Lists
6. Print Lists
0. Exit
Enter your choice: 3
List 1 sorted.
Menu

1. Insert into List 1
2. Insert into List 2
3. Sort List 1
4. Reverse List 2
5. Concatenate Lists
6. Print Lists
0. Exit
Enter your choice: 6
List 1: 2 3 9
List 2:
Menu

1. Insert into List 1
2. Insert into List 2
3. Sort List 1
4. Reverse List 2
5. Concatenate Lists
6. Print Lists
0. Exit
Enter your choice: 4
List 1 reversed.
```

```
Menu

1. Insert into List 1
2. Insert into List 2
3. Sort List 1
4. Reverse List 2
5. Concatenate Lists
6. Print Lists
0. Exit
Enter your choice: 5
Lists concatenated.
Menu

1. Insert into List 1
2. Insert into List 2
3. Sort List 1
4. Reverse List 2
5. Concatenate Lists
6. Print Lists
0. Exit
Enter your choice: 2
Enter data to insert into List 2: 45
Menu

1. Insert into List 1
2. Insert into List 2
3. Sort List 1
4. Reverse List 2
5. Concatenate Lists
6. Print Lists
0. Exit
Enter your choice: 5
Lists concatenated.
```

```
Menu

1. Insert into List 1
2. Insert into List 2
3. Sort List 1
4. Reverse List 2
5. Concatenate Lists
6. Print Lists
0. Exit
Enter your choice: 6
List 1: 9 3 2 45
List 2: 45
Menu

1. Insert into List 1
2. Insert into List 2
3. Sort List 1
4. Reverse List 2
5. Concatenate Lists
6. Print Lists
0. Exit
Enter your choice: 7
```

2 a) Stack implementation using single linked list

ans)

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
    int data;
    struct node *next;
} *top, *top1, *temp;

int count = 0;
void push (int value)
{
    if (top == NULL)
    {
        top = (struct node *) malloc (1*sizeof (struct node));
        top -> next = NULL;
        top -> data = value;
    }
    else
    {
        temp = (struct node *) malloc (1* sizeof (struct node));
        temp -> next = top;
        temp -> data = value;
        top = temp;
    }
    count ++;
    printf (" Node inserted \n");
}

int pop()
{
```

SLL - Stacks, Queues

Ext

29/11/24

```
    top1 = top;
    if (top1 = NULL)
    {
        printf("\n stack underflow");
        return -1;
    }
    else {
        top1 = top -> next
        }
        int popped = top -> data
        free (top);
        top = top1;
        count--;
        return popped;
}

void display()
{
    top1 = top;
    if (top1 = NULL)
    {
        printf("\n stack underflow");
        return;
    }
    printf("The stack is :");
    while (top1 != NULL)
    {
        printf("%d\t", top1 -> data);
        top1 = top1 -> next;
    }
}

void main()
{
```

```
int choice, value;
while (1)
{
    printf("\nmenu \n 1. Push \t 2. Pop\t 3. Display
    4. Exit \n");
    printf("Enter your choice:");
    scanf("%d", &choice);
    switch (choice)
    {
        case 1:
            printf("Enter value:");
            scanf("%d", &value);
            push(value);
            break;
        case 2:
            printf("Popped element: %d", pop());
            break;
        case 3:
            display();
            break;
        case 4:
            exit(0);
            break;
        default:
            printf("Invalid choice");
    }
}
}
```

OUTPUT:

```
Menu
1. Push    2. Pop    3. Display    4. Exit
Enter your choice: 1
Enter value: 2
```

Menu

1. Push 2. Pop 3. Display 4. Exit

Enter your choice : 1

Enter value : 4

Menu

1. Push 2. Pop 3. Display 4. Exit

Enter your choice : 2

Popped element : 4

Menu

1. Push 2. Pop 3. Display 4. Exit

Enter your choice : 3

The stack is : 2

Menu

1. Push 2. Pop 3. Display 4. Exit

Enter your choice : 4

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

Enter the value to insert: 3
Node is Inserted

Menu
1. Push 2. Pop  3. Display      4. Exit
Enter your choice : 1

Enter the value to insert: 8
Node is Inserted

Menu
1. Push 2. Pop  3. Display      4. Exit
Enter your choice : 2
Popped element is :8
Menu
1. Push 2. Pop  3. Display      4. Exit
Enter your choice : 3
The stack is: 3
Menu
1. Push 2. Pop  3. Display      4. Exit
Enter your choice : 4
```

2 b) Queue implementation using Single linked list

ans)

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
    int data;
    struct node * next;
};
struct node * front = NULL;
struct node * rear = NULL;

void enqueue (int value)
{
    struct node *ptr;
    ptr = (struct node *) malloc (sizeof (struct node));
    ptr -> data = value;
    ptr -> next = NULL;
    if ((front == NULL) && (rear == NULL))
    {
        front = rear = ptr;
    }
    else
    {
        rear -> next = ptr;
        rear = ptr;
    }
    printf (" Node inserted ");
}

int dequeue ()
{
    if (front == NULL)
    {
```

```
        printf("\n Underflow");
        return -1;
    }
    else
    {
        struct node * temp = front;
        int temp_data = front -> data;
        front = front -> next;
        free(temp);
        return temp_data;
    }
}

void display()
{
    struct node * temp;
    if ((front == NULL) && (rear == NULL))
    {
        printf("Queue is empty");
    }
    else
    {
        printf("The queue is :");
        temp = front;
        while (temp != NULL)
        {
            printf("%d\t", temp -> data);
            temp = temp -> next;
        }
        printf("\n");
    }
}

void main()
{
```

```
int choice, value;
while (1)
{
    printf("\n Menu \n 1. Enqueue \t 2. Dequeue \t
    3. Display \t 4. Exit \n");
    printf("Enter your choice:");
    scanf("%d", &choice);
    switch (choice)
    {
        case 1:
            printf("Enter value to be inserted");
            scanf("%d", &value);
            enqueue(value);
            break;
        case 2:
            printf("Element removed: %d",
            dequeue());
            break;
        case 3:
            display();
            break;
        case 4:
            exit(0);
            break;
        default:
            printf("Invalid choice");
    }
}
}
```

OUTPUT:

Menu

1. Enqueue 2. Dequeue 3. Display 4. Exit

Enter your choice: 1

Enter value to be inserted : 2

Menu

1. Enqueue 2. Dequeue 3. Display 4. Exit

Enter your choice : 1

Enter value to be inserted : 8

Menu

1. Enqueue 2. Dequeue 3. Display 4. Exit

Enter your choice : 2

Element removed : 2

Menu

1. Enqueue 2. Dequeue 3. Display 4. Exit

Enter your choice : 3

The queue is : 8

Menu

1. Enqueue 2. Dequeue 3. Display 4. Exit

Enter your choice : 4

```
Menu
1.Enqueue       2.Dequeue       3.Display       4.Exit

Enter your choice : 1

Enter the value to insert: 2
Node is Inserted

Menu
1.Enqueue       2.Dequeue       3.Display       4.Exit

Enter your choice : 1

Enter the value to insert: 8
Node is Inserted

Menu
1.Enqueue       2.Dequeue       3.Display       4.Exit

Enter your choice : 2
Element removed is :2
Menu
1.Enqueue       2.Dequeue       3.Display       4.Exit

Enter your choice : 3
The queue is
8       Menu
1.Enqueue       2.Dequeue       3.Display       4.Exit

Enter your choice : 4
```