# LAB - 7, 8

```
#include <stdio.h>
#include <stdlib.h>
Struct node {
    int data;
    struct node * next;
};
Struct node *rear = NULL, *front = NULL, *top = Null;
Struct node ~~*rear = NULL~~ * getnode( int item) {
    Struct node *newn = (struct node *)
                        malloc( sizeof(struct node
    newn→data = item;
    newn→next = NULL;
    return newn;
}
void display ( struct node *head) {
    if ( head == NULL) {
        printf (" List is empty \n");
        return;
    }
struct node * ptr = head;
while ( ptr) {
        printf ("%d →", ptr→data);
```

```
        ptr = ptr -> next;
    }
    printf("\b \b\b \n");
}
Struct node* insertfront(struct node * head,
                         int item)
{
    struct node * newn = getnode(item);
    newn -> next = head;
    head = newn;
    return head;
}
void swap(int *a, int *b) {
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}
struct node *Sort(struct node * head)
{
    int sorted;
    if (head == NULL) return head;
    struct node * ptr = head;
    do {
```

```
{
    ptr = head;
    sorted = 0;
    while ( ptr -> next) {
        if( ptr -> data > ptr -> next -> data)
        {
            swap ( & ptr -> data, & ptr -> next
                                    -> data);

            sorted = 1;
        }
        ptr = ptr -> next;
    }
    } while (sorted == 1);
    return head;
}
void reverse ( struct node ** head)
{
    Struct node * prev = NULL;
    Struct node * current = * head;
    Struct node * next = NULL;
    while (current != NULL){
        next = current -> next;
        current -> next = prev;
        prev = current;
        current = next;
}   * head = prev;
}
```

```
struct node * concatenate (struct node * head1,
                           struct node * head2)
{
    struct node * ptr = head1;
    while ( ptr -> next)
    {
        ptr = ptr -> next;
    }
    ptr -> next = head2;
    return head1;
}

void qinsert () {
    struct node * newnode;
    newnode = (struct node *) malloc(
                    sizeof (struct node));
    printf("Enter the element : \n");
    scanf(" %d", &newnode -> data);
    newnode -> next = NULL;
    if (rear == NULL) {
        rear = newnode;
        front = newnode;
    }
    else {
```

```
        rear -> next = newnode;
        rear = newnode;
    }
}
void qdel() {
    if (front == NULL) {
        printf("Queue is empty \n"); return;
    }
    else {
        printf("Deleted ele is %d", front->data);
        if (front == rear) {
            printf("Queue is empty \n");
            front = NULL; rear = NULL;
        }
        else
            front = front -> next;
    }
}
void qdisplay() {
    struct node *temp;
    if (front == NULL) {
        printf("Queue is empty");
        return;
    }
```

```
    temp = front;
    while ( temp != NULL) {
        printf(" %d", temp -> data);
        temp = temp -> next;

    }

}

void spush(){

    int item;
    Struct node *newnode;
    printf(" Enter the element \n");
    scanf(" %d", &item);
    newnode =( struct node*) malloc(
                        sizeof( struct node));
    newnode -> data = item;
    newnode -> next = NULL;
    if (top == NULL)
        top = newnode;
    else
        newnode -> next = top;
        top = newnode;
}
void spop(){
```

```
    if ( top == NULL)
        printf(" Stack is empty !! ");

    else {
        printf(" Element removed is %d :", top->dat
        top = top -> next;

    }
}
void sdisplay() {
    struct node *temp;
    temp = top;
    if( top == NULL)
        printf (" Stack is empty ");
    while (temp != NULL) {
        printf (" %d", temp -> data);
        printf ("\n");
        temp = temp -> next;

    }
}
int main () {
    printf(" Linked list program containing
                sort, reverse, concatenate func.\n");
    int n1, n2, n, ch, flag=0;
    int choice;
```

```
struct node *head1 = NULL, *head2 = NULL;
do {
    printf("\nEnter choice : \n 1. Stack \n2. Queue
        \n3. Linked list1 \n 4. Linked list2 \n
                                5. Exit \n");
    scanf("%d", &n1);
    switch(n1) {
        case 1: {
                do {
                    printf("\n1. Push \n2. Display
                            \n3. Pop \n");
                    printf("Enter your choice");
                    scanf("%d", &choice);
                    switch( choice) {
                        case 1: spush(); break;
                        case 2: sdisplay(); break;
                        case 3: spop(); break;
                    }
                } while (choice != 10);
    }
    case 2: {
                do {
                    printf("\nQueue implementation using linked
                                list \n");
```

```
printf("\n1. Create\n2. Display \n3. Delete \n4 Exit ");
printf("Enter your choice: ");
scanf("%d", &choice);
switch (choice) {
    case 1: qinsert(); break;
    case 2: qdisplay(); break;
    case 3: qdel(); break;
}
} while( choice != 10);
}
case 3: {
    do{
        printf("3:Insert \n4: Sort \n5:
                Revrse \n6: Concatinate
with list \n7: Display list \n8:
    to back to main menue \n9: Exit");
scanf ("%d", &n2);
switch( n2){
    {
    Case 3: {
        printf("Enter item to be entered:");
        scanf("%d", &n);
        head = insertfront (head, n);
        break;
    }
```

```
Case 4: {
        Head1 = sort (Head1);
        break;
    }
Case 5: {
        reverse (&Head1);
        break;
    }
Case 6: {
        Head1 = Concatenate (Head1, Head2);
        break;
    }
Case 7: {
        display (Head1);
        break;
    }
Case 8: {
        flag = 1;
        break;
    }
Case 9: {
        exit(0);
    }
default: printf("Invalid input
                \n");
}
```

```
if (flag == 1) {
        break;

    }
} while(1);
  break;
}
Case 4: {
      flag = 0;
       do{
        printf(" 3: Insert \n4: Sort \n5: Reverse
            \n6: Concatinate with list1 \n
            7: Display list \n8. Go back
              \n9: Exit \n");

        scanf("%d", &n2);
        switch( n2)
        {
          case 3: {
                printf(" Enter item to be
                          inserted ");
                scanf("%d",&n);
                head2 = insatfront(head2,n);
                break;
            }
```

```
case 4: {
        head2 = sort(head2);
        break;
    }
case 5: {
        reverse(&head2);
        break;
    }
case 6: {
        head2 = concatenate(head2,
                            head1);
        break;
    }
case 7: {
        display(head2);
        break;
    }
case 8: {
        flag = 1;
        break;
    }
case 9: {
        exit(0);
    }
default: printf("Invalid input\n");
}
if(flag == 1){
```

```
            flag = 0; break;
        } while(1);

        break;

    }
    case 9: exit(0);
    default: printf("Invalid Input !\n");
        }
    } while(1);
    return 0;
}
```