Lab 7

```
#include <stdio.h>
#include <stdlib.h>
    struct node {
            int data;
            struct node * next;
            };
        struct node *head;
        struct node * head2;
void push() {
            struct node *ptr;
            int new_data;
            ptr = (struct node *) malloc (size of (struct node));
            if (ptr == NULL)
                {
                    printf ("Overflow \n");
                }
            else { printf ("Enter the value to be
                            inserted: ");
            scanf ("%d", &new_data);
            ptr -> data = new data;
            ptr -> next = head;
              head = ptr;
            phintf ("Nocle inserted at top of stack\n");
            }
            }
void enqueue () {
            struct node *ptr, *temp;
            int new_data;
            ptr = (struct node *) malloc (size of (struct node));
          printf ("Enter value to be inserted");
           scanf ("%d", &new_data);
          ptr -> data = new_data;
```

```
if (head == NULL)
    {
      ptr -> next = NULL;
       head = ptr;
      printf("Node inserted at rear Q");
      else
        {
          temp = head;
          while (temp -> next! = NULL) {
                    temp = temp -> next; }
          temp -> next = ptr;
          ptr -> next = NULL;
        printf("Node inserted at rear of Q"); }}

void dequeue() {
        struct node * ptr;
    if (head == NULL)
      {
        printf("Empty"); }
      else
        {
          ptr = head;
        head = ptr -> next;
          free (ptr);
        printf("NODE DELETED FROM FRONT OF QUEUE)
            }}

void display() {
        struct node * ptr;
          ptr = head;
      if (ptr == NULL) { printf("Empty \n"); }
```

```
else { printf ("\n\n List -> ");
       while (ptr1 = NULL) {
            printf (ptr1 = NULL) {

                printf ("%d", ptr -> data);
                ptr = ptr -> next;
                }}}

void sort () {
    struct node * ptr = head;
    struct node *temp = NULL;
     int i ,
    if ( head == NULL ) {
                        return;
                        }

    else { while (ptr1 = NULL) {
                    temp = ptr -> next;
                    while (temp1 = NULL ) {
                if (ptr -> data -> temp -> data) {
                            i = ptr -> data;
                ptr -> data = temp -> data;
                  temp -> data = ii
                    } temp = temp -> next;
                  } ptr = ptr -> next;
                        }}}

void reverse() {
    struct node * prev = NULL;
    struct node * next = NULL;
    struct node * ptr = head;
```

```
while (ptr1 != NULL) {
    next = ptr -> next;
    ptr -> next = prev; prev = ptr;
    prev = next
    }
    head = prev;
    }

struct node * create_list (struct node *head){
        struct node, *ptr, *temp;
        int i, n, new_data;
    printf ("Enter number of nodes: ");
    scanf ("%d", &n);
    node = NULL;
    if (n==0) { return head; }
    for (i=1; i<=n; i++)
        {
        ptr = (struct node *) malloc (sizeof (struct node));
    printf ("Enter the element to be inserted: ");
    scanf ("%d", &new_data);
    if (head == NULL)
        {
        ptr -> next = NULL;
        head = ptr;
        }
```

```
else { temp = head;
    while (temp -> next1 = NULL)
    {
        temp = temp ->next;
        }

    temp -> next = ptr;
    ptr -> next = NULL; }}}

struct node *concatenated ( struct node *head, struct node)
    {
      struct node *ptr;
      if ( head == null )
        {
            head = head2;
              Return head; }

    if (head2 == NULL ) {
            return head; }
       ptr = head;
    while (ptr -> next1 = NULL) {
            ptr = ptr -> next; }

    int main() { int charie = 0;
      while()) {
        printf("1. Push 2. Pop 3 Display 4
                Enqueue 4 DeQ 5 DisP)";
       switch()
          {
          case :
          }
```

Lab 8

```
struct node {
        int data ;
        struct node *next;
        struct node *prev;
        };
struct node * head = = NULL;
void insert_beg()
  {
    struct node * new_node ;
     new_node -> next = NULL
     new_node -> prev = NULL;
       if (head == NULL) {
                head = new_node;
                }
          else
            {
             new_node -> next = head ;
             head -> prev = new_node; }
             head = new_node; } }

void insert_end () {
                struct node * new_node, *temp;
new_node = (struct node*) malloc (sizeof (structnode));
   new_node -> next = NULL;
   new_node -> prev = NULL;
  if (node = NULL) { head = new_node ;}
```

```
else { temp = head;
       while (temp -> next != NULL)
       temp = temp -> next;
       temp -> next = zero_node;
       new_node -> prev = temp; }}

void insert_between () {
     if (head == null) { "empty"; } {
        while (temp -> data1 = ele) {
              temp = temp -> next;
           if (temp == NULL) {
                 printf ("Element is not
                          in list");
           }

new_node -> next = temp -> next
     temp -> next; nexnode;
     if (temp == NULL ) {
           printf ("Element is not in list");
           }

new_nod
```

```
void del() {
    if(empty());

    while (temp -> data! = ele) {

        temp - tem -> next;
        if (temp == NULL)
            {
                printf(" Element not in list \n");
                break;
            }
        if(temp == head) { head => head -> next; }
        else if (temp -> next = NULL)
                {
                    temp = temp -> prev;
                    temp => next = NULL;
                }
        else {
            temp -> prev -> next = temp -> next;
            temp -> next -> prev = temp -> prev;
            }

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