LAB NO 7:

Q1.

#include <stdio.h> #include <stdlib.h> typedef struct node {

int data;

struct node\* next;

}\*NODE;

NODE enqueue (NODE first, int e) {

NODE temp=(NODE)malloc(sizeof(struct node)); temp->data=e;

temp->next=NULL; if (first==NULL) {

return temp;

}

else {

NODE m=first;

while(m->next!=NULL) { m=m->next;

}

m->next=temp; return first;

}

}

NODE dequeue(NODE first) {

if (first==NULL) { printf("Queue Empty\n");

return NULL;

}

else if (first->next==NULL) { printf("Dequeue:\t%d\n",first->data); free(first);

return NULL;

}

else {

}

}

NODE temp=first; first=first->next;

printf("Dequeue:\t%d\n",temp->data); free(temp);

return first;

void display(NODE first) { if (first==NULL) {

printf("Queue Empty\n");

}

else {

NODE p=first;

while (p->next!=NULL) {

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

}

printf("%d\n",p->data);

}

}

int main() {

NODE first=NULL; int z=0,k;

while (z==0) {

printf("1. Enqueue\t2. Dequeue\t3. Display\t4. Exit\t"); scanf("%d",&k);

switch(k) {

case 1: {

int i; printf("Enqueue:\t"); scanf("%d",&i); first=enqueue(first,i); break;

}

case 2: {

first=dequeue(first); break;

}

case 3: {

display(first); break;

}

case 4: {

z=1;

break;

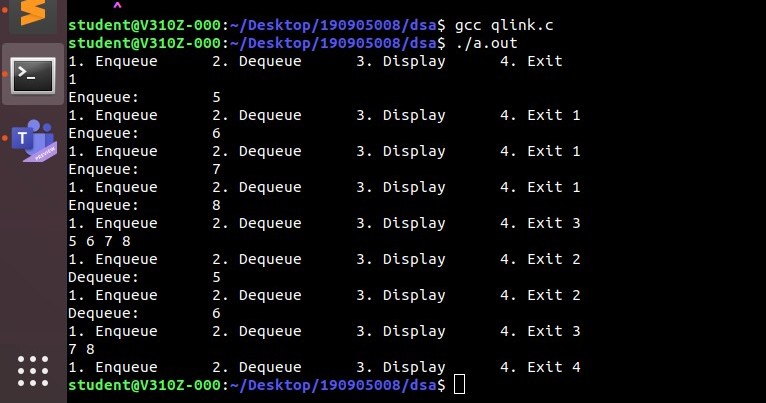
}

}

}

return 0;

}



Q2.

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

struct node

{

int num;

struct node \*next;

};

void create(struct node \*\*);

void findunion(struct node \*, struct node \*, struct node \*\*); void findintersect(struct node \*, struct node \*, struct node \*\*); void display(struct node \*);

void release(struct node \*\*);

int main()

{

struct node \*phead, \*qhead, \*intersect, \*unionlist;

phead = qhead = intersect = unionlist = NULL; printf("Enter elements in the list 1\n"); create(&phead);

printf("\nEnter elements in the list 2\n"); create(&qhead);

findunion(phead, qhead, &unionlist); findintersect(phead, qhead, &intersect); printf("\nDisplaying list 1:\n"); display(phead);

printf("Displaying list 2:\n"); display(qhead);

printf("Displaying the union of the 2 lists:\n"); display(unionlist);

printf("Displaying the intersection of the 2 lists:\n"); if (intersect == NULL)

{

printf("Null\n");

}

else

{

display(intersect);

}

release(&phead); release(&qhead); release(&unionlist); release(&intersect);

return 0;

}

void findintersect(struct node \*p, struct node \*q, struct node \*\*intersect)

{

struct node \*ptemp, \*qtemp, \*itemp, \*irear, \*ifront;

ptemp = p;

while (ptemp != NULL)

{

qtemp = q;

ifront = \*intersect;

while (qtemp != NULL && ptemp->num != qtemp->num)

{

qtemp = qtemp->next;

}

if (qtemp != NULL)

{

if (ifront != NULL)

{

if (ifront->num == qtemp->num)

{

ptemp = ptemp->next; continue;

}

ifront = ifront->next;

}

itemp = (struct node \*)malloc(sizeof(struct node)); itemp->num = qtemp->num;

itemp->next = NULL; if (\*intersect == NULL)

{

\*intersect = itemp;

}

else

{

irear->next = itemp;

}

irear = itemp;

}

ptemp = ptemp->next;

}

}

void findunion(struct node \*p, struct node \*q, struct node \*\*unionlist)

{

struct node \*utemp, \*ufront, \*urear; int flag = 0;

while (p != NULL)

{

ufront = \*unionlist; while (ufront != NULL)

{

if (ufront->num == p->num)

{

flag = 1;

}

ufront = ufront->next;

}

if (flag)

{

flag = 0;

}

else

{

utemp = (struct node \*)malloc(sizeof(struct node)); utemp->num = p->num;

utemp->next = NULL; if (\*unionlist == NULL)

{

\*unionlist = utemp;

}

else

{

urear->next = utemp;

}

urear = utemp;

}

p = p->next;

}

while (q != NULL)

{

ufront = \*unionlist; while (ufront != NULL)

{

if (ufront->num == q->num)

{

flag = 1;

}

ufront = ufront->next;

}

if (flag)

{

flag = 0;

}

else

{

utemp = (struct node \*)malloc(sizeof(struct node)); utemp->num = q->num;

utemp->next = NULL; if (\*unionlist == NULL)

{

\*unionlist = utemp;

}

else

{

urear->next = utemp;

}

urear = utemp;

}

q = q->next;

}

}

void create(struct node \*\*head)

{

struct node \*temp, \*rear; int ch, a;

do

{

printf("Enter a number: "); scanf("%d", &a);

temp = (struct node \*)malloc(sizeof(struct node)); temp->num = a;

temp->next = NULL; if (\*head == NULL)

{

\*head = temp;

}

else

{

rear->next = temp;

}

rear = temp;

printf("Do you want to continue [1/0] ? "); scanf("%d", &ch);

} while (ch != 0);

}

void display(struct node \*head)

{

while (head != NULL)

{

printf("%d ", head->num); head = head->next;

}

printf("\n");

}

void release(struct node \*\*head)

{

struct node \*temp = \*head; while ((\*head) != NULL)

{

(\*head) = (\*head)->next; free (temp);

temp = \*head;

}

}

