Lab Session 4:

4.1 : Queue

#include <stdio.h>

#include <stdlib.h>

//implement dynamic data structures

struct node {

int value;

struct node \* next;

};

/\*

Implementation of basic queue using double pointer

enqueue

dequeue

peek

display

\*/

void enqueue(struct node \*\* front,struct node \*\*rear,int val){

//enqueue means adding to queue

struct node \* t = (struct node \*)malloc(sizeof(struct node));

if (!t)return;

t->value = val;

t->next = NULL;

if ((\*front == \*rear) && (\*front == NULL)){

\*front = t;

\*rear = t;

}else {

(\*rear)->next = t;

\*rear = t;

}

}

struct node \* dequeue(struct node \*\* front,struct node \*\* rear){

//check NULL

struct node \* t = NULL;

if (!(\*front))

return t;

t = \*front;

if (\*front == \*rear){

\*front = NULL;

\*rear = NULL;

t->next = NULL;

return t;

}

\*front = (\*front)->next;

t->next = NULL;

return t;

}

struct node \* peek(struct node \*\*front){

struct node \* p = NULL;

if (\*front){

p = (struct node \*)malloc(sizeof(struct node));

p->value = (\*front)->value;

p->next = NULL;

}

return p;

}

void display(struct node \*\* front){

struct node \* p;

printf("[ ");

for (p = \*front;p!=NULL;p = p->next)

printf("%d ",p->value);

printf(" ]\n");

}

// void display(struct node \*\* start){}

int main(){

struct node \* f = NULL,\*r = NULL,\*q,\*w,\*e;

enqueue(&f,&r,1);

enqueue(&f,&r,2);

enqueue(&f,&r,3);

enqueue(&f,&r,4);

q = dequeue(&f,&r);

w = peek(&f);

printf("Q is %d\n",q->value);

display(&f);

return 0;

}