Week5:

Q1.

#include <stdio.h> #include <string.h> #include <stdlib.h> typedef struct

{

char arr[5]; int front; int rear;

int count;

}Queue;

void insertq(Queue \*q,char ele)

{

if(q->count==5) printf("QUEUE FULL\n"); else

{

q->rear=(q->rear+1)%5; q->arr[q->rear]=ele;

q->count++;

}

}

void deleteq(Queue \*q)

{

if(q->count==0) printf("QUEUE EMPTY\n"); else

{

q->front=(q->front+1)%5; q->count--;

}

}

void displayq(Queue \*q)

{

if(q->count==0) printf("QUEUE EMPTY\n"); else

{

for(int i=1,f=q->front;i<=q->count;i++)

{

printf("%c\n",q->arr[f] ); f=(f+1)%5;

}

}

}

void main()

{

Queue q1,\*q; q=&q1;

q->front=0; q->rear=-1; int j,ch;

q->count=0; char ele; for(;;)

{

printf("Enter choice 1)INSERT 2)DELETE 3)DISPLAY 4)EXIT\n"); scanf("%d",&ch);

if(ch==1)

{

printf("Enter char\n"); scanf(" %c",&ele); insertq(q,ele);

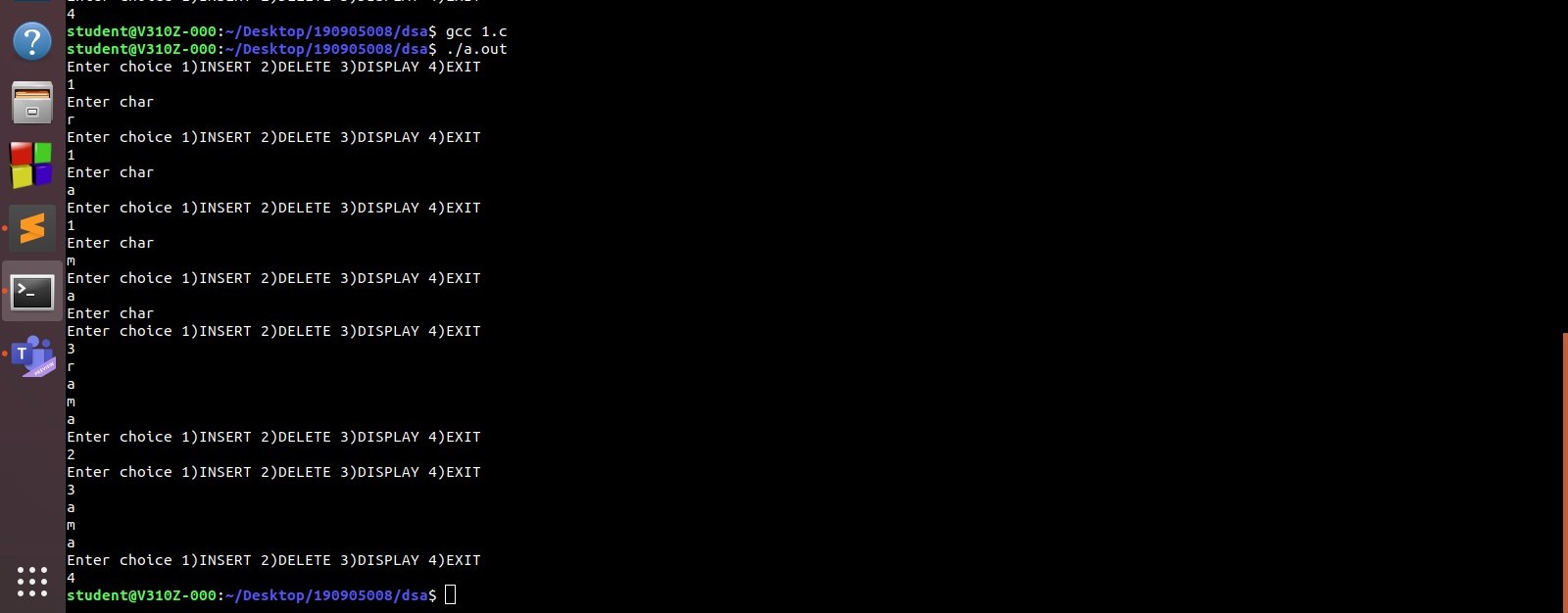
}

else if(ch==2) deleteq(q); else if(ch==3) displayq(q); else

exit(0);

}

}



Q2.

#include <stdio.h> #include <stdlib.h> #define SIZE 10

#define UNDERFLOW\_INT -32767

typedef enum { NO = 0,

YES = 1,

} BOOL;

typedef struct CircularQueue { int \* arr;

int front1, rear1, cap1; int front2, rear2, cap2;

} CQUEUE\_t;

typedef CQUEUE\_t \* CQUEUE\_p\_t;

BOOL isFullQueue (CQUEUE\_t queue, int qno) { if (qno == 1 && queue.cap1 == SIZE/2)

return YES;

else if (qno == 2 && queue.cap2 == SIZE/2) return YES;

return NO;

}

BOOL isEmptyQueue (CQUEUE\_t queue, int qno) { if (qno == 1 && queue.cap1 == 0)

return YES;

else if (qno == 2 && queue.cap2 == 0) return YES;

return NO;

}

void insert (CQUEUE\_p\_t queue, int item, int qno) { if (isFullQueue(\*queue, qno)) {

printf("\n\t\tQUEUE '%d' OVERFLOW!\n\n", qno); return;

}

if (qno == 1) {

if (isEmptyQueue(\*queue, qno)) queue->front1 = queue->rear1 = 0; else if (queue->rear1 == SIZE/2 - 1) queue->rear1 = 0;

else

queue->rear1 += 1;

\*(queue->arr + queue->rear1) = item; queue->cap1++;

}

if (qno == 2) {

if (isEmptyQueue(\*queue, qno))

queue->front2 = queue->rear2 = SIZE - 1; else if (queue->rear2 == SIZE/2)

queue->rear2 = SIZE - 1;

else

queue->rear2 -= 1;

\*(queue->arr + queue->rear2) = item; queue->cap2++;

}

}

int delete (CQUEUE\_p\_t queue, int qno) { if (isEmptyQueue(\*queue, qno)) {

printf("\n\t\tQUEUE '%d' UNDERFLOW!\n\n", qno); return UNDERFLOW\_INT;

}

int item = 0;

if (qno == 1) {

item = \*(queue->arr + queue->front1);

\*(queue->arr + queue->front1) = 0; if (queue->front1 == queue->rear1) queue->front1 = queue->rear1 = -1; else if (queue->front1 == SIZE/2 - 1) queue->front1 = 0;

else

queue->front1 += 1; queue->cap1--;

}

if (qno == 2) {

item = \*(queue->arr + queue->front2);

\*(queue->arr + queue->front2) = 0; if (queue->front2 == queue->rear2)

queue->front2 = queue->rear2 = SIZE - 1; else if (queue->front2 == SIZE/2)

queue->front2 = SIZE - 1; else

queue->front2 -= 1; queue->cap2--;

}

return item;

}

void display (CQUEUE\_t queue, int qno) { if (isEmptyQueue(queue, qno)) {

printf("\n\t\tEMPTY QUEUE %d.\n\n", qno); return;

}

printf("\n\tQUEUE '%d': ", qno); int i;

if (qno == 1) {

if (queue.rear1 >= queue.front1)

for (i = queue.front1; i <= queue.rear1; ++i) printf("\t%d", \*(queue.arr + i));

else {

for (i = queue.front1; i < SIZE/2; ++i) printf("\t%d", \*(queue.arr + i));

for (i = 0; i <= queue.rear1; ++i) printf("\t%d", \*(queue.arr + i));

}

}

else if (qno == 2) {

if (queue.rear2 <= queue.front2)

for (i = queue.front2; i >= queue.rear2; --i) printf("\t%d", \*(queue.arr + i));

else {

for (i = queue.front2; i >= SIZE/2; --i) printf("\t%d", \*(queue.arr + i));

for (i = SIZE - 1; i >= queue.rear2; --i) printf("\t%d", \*(queue.arr + i));

}

}

printf ("\n\n");

}

int main(int argc, const char \* argv[]) {

printf("\n\n This program implements two circular queues in a single array.\n\n"); CQUEUE\_p\_t queue = (CQUEUE\_p\_t)malloc(sizeof(CQUEUE\_t));

queue->arr = (int \*)calloc(SIZE, sizeof(int)); queue->front1 = queue->rear1 = -1;

queue->front2 = queue->rear2 = SIZE; queue->cap1 = queue->cap2 = 0;

int qno; do {

printf("\n\n------------------main option----------------------\n 1. Queue no. 1.\n 2. Queue no. 2.\n 3. Display Both queue1 and queue2.\n Any other key for exit.\n\n Enter your choice: "); scanf("%d", &qno);

if (qno == 3) { display(\*queue, 1);

display(\*queue, 2); continue;

}

else if (!(qno == 1 || qno == 2)) exit(6);

printf("\n\t| You have choosen Queue '%d'.\n", qno); char choice;

do {

printf("\n\t| 1. Insert.\n\t| 2. Delete.\n\t| 3. Display.\n\t| Anything else to go back.\n\t| Enter choice: ");

scanf(" %c", &choice); if (choice == '1') {

int item;

printf("\n\t| Enter item to insert: "); scanf("%d", &item);

insert(queue, item, qno);

}

else if (choice == '2') {

int item = delete(queue, qno);

if (item != UNDERFLOW\_INT) printf("\n\t| Deleted Item = %d.\n", item);

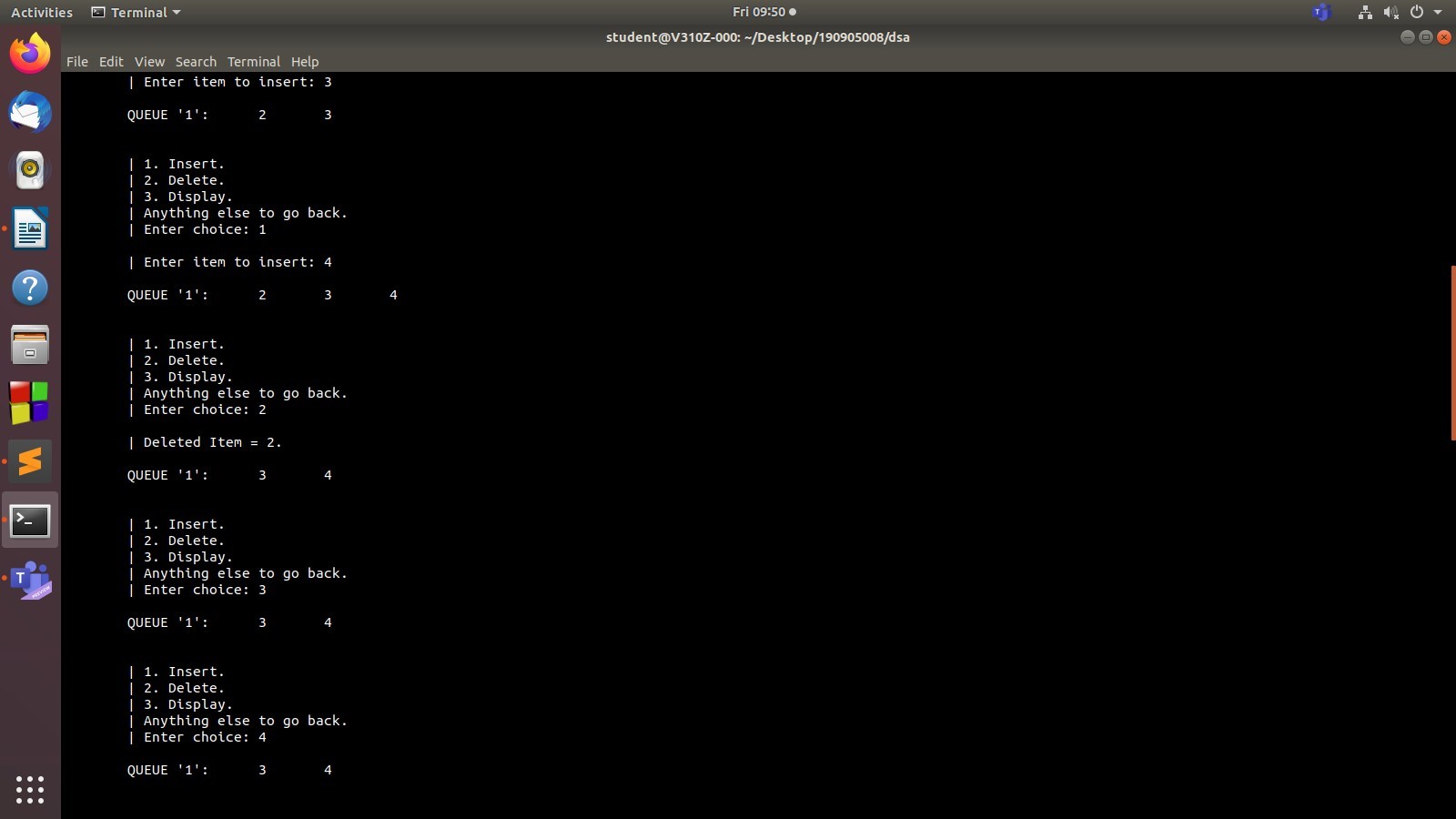
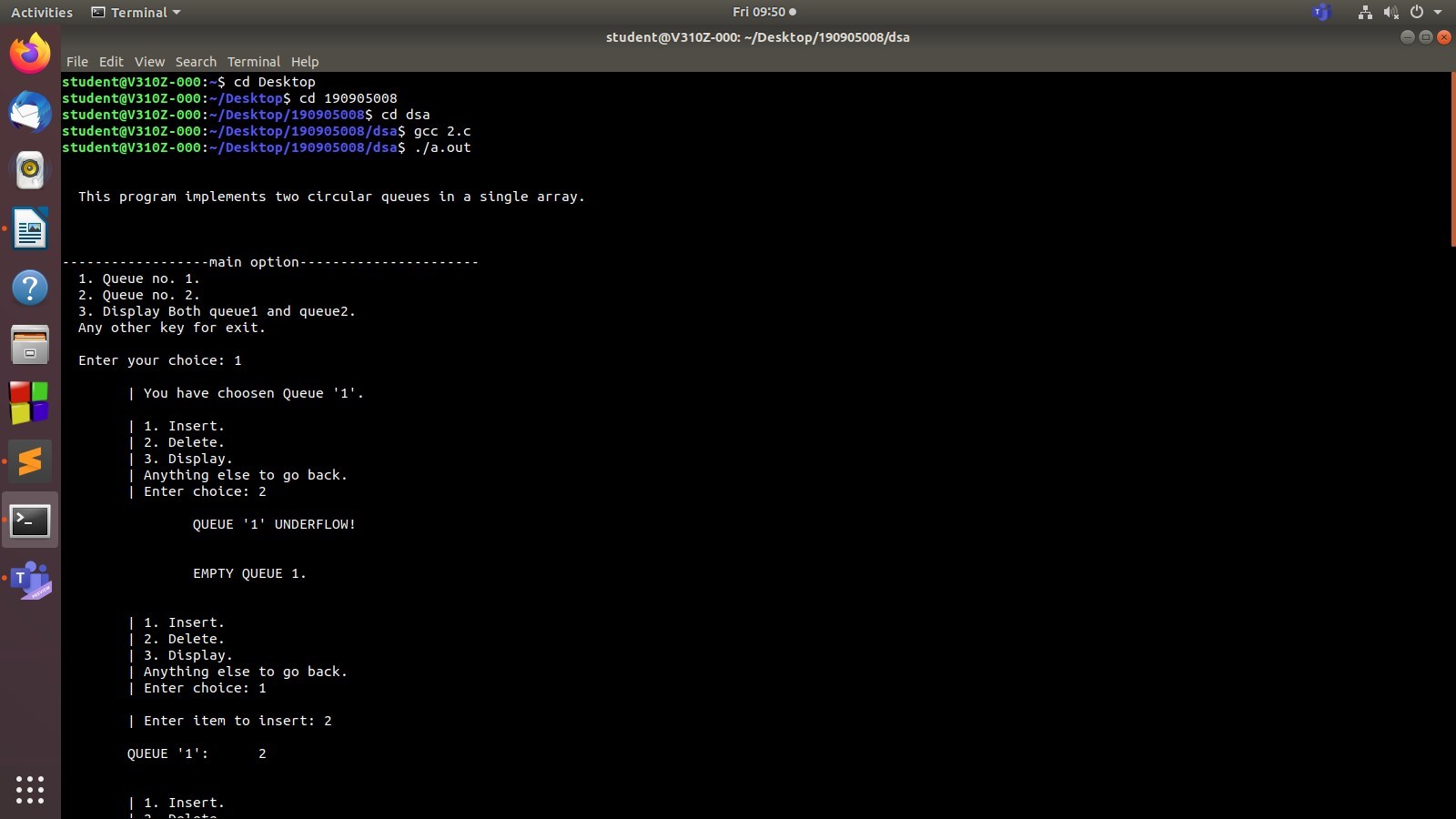
}

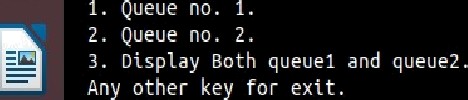
display(\*queue, qno);

} while (choice == '1' || choice == '2' || choice == '3');

} while (qno == 1 || qno == 2 || qno == 3);

}



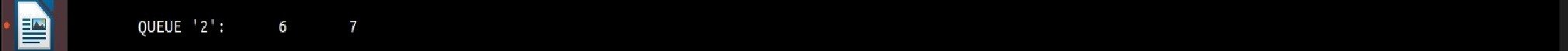
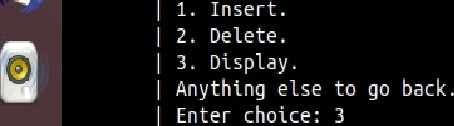


studcn£@ V310 Z •000: -/Dcsktop/19090S008/dsa



s£u den £@ V310 Z •000: -/D eskt op/190905008/ds a

1. I n s ert .



3. Disptay Both qseuel and qsese2.



3. Dtsptay Both qteuel and qtete2.



Q3.

#include <stdio.h> #include <stdlib.h> #define size 100

struct stack{

int data[size]; int top;

}stackin, stackout;

void push(struct stack \*st, int num){ st->top++;

st->data[st->top] = num;

}

int pop(struct stack \*st){

int popped = st->data[st->top]; st->top--;

return popped;

}

void pushqueue(int data){ push(&stackin, data);

}

int popqueue(){ if(stackout.top == -1){

while(stackin.top != -1){ push(&stackout, pop(&stackin));

}

}

int num = pop(&stackout); return num;

}

void displayqueue(){

for(int a = stackout.top; a >= 0; a--){ printf("%d ", stackout.data[a]);

}

for(int b = 0; b <= stackin.top; b++){ printf("%d ", stackin.data[b]);

}

printf("\n\n");

}

int main()

{

stackin.top = stackout.top = -1; int choice = 0, num;

while (1) {

printf("Type 1 to insert, 2 to delete, 3 to display and 4 to exit.\n"); printf("Enter your choice: ");

scanf("%d", &choice); switch (choice){

case 1:

printf("Enter a number to insert: "); scanf("%d", &num); pushqueue(num);

printf("%d is inserted!\n\n",num); break;

case 2:

if(stackin.top == -1 && stackout.top == -1){ printf("The queue is empty!\n\n");

}

else{

num = popqueue();

printf("%d is deleted.\n\n", num);

}

break; case 3:

if(stackin.top == -1 && stackout.top == -1){ printf("The queue is empty!\n\n");

}

else{ displayqueue();

}

break; case 4:

exit(0); default:

printf("invalid\n\n");

}

}

return 0;

}

