LAB TASK\_5

#include<stdio.h>

#define SIZE 5

void enQueue(int value);

void deQueue();

void display();

void peek();

int queue[10], front = -1, rear = -1;

int value, choice;

int main() {

while (1) {

printf("\n MENU\n");

printf("1.enQueue 2.deQueue 3.display 4.peek 5.exit");

printf("\nenter the choice:");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("enter the value to be insert:");

scanf("%d", &value);

enQueue(value);

break;

case 2:

deQueue();

break;

case 3:

display();

break;

case 4:

peek();

break;

case 5:

exit(0);

default:

printf("enter the valid choice");

}

}

}

void enQueue(int value) {

if (rear == SIZE - 1) {

printf("queue is full");

} else if (front == -1 && rear == -1) {

front = rear = 0;

queue[rear] = value;

} else {

rear++;

queue[rear] = value;

}

}

void deQueue() {

if (front == -1 && rear == -1) {

printf("queue is empty");

} else if (front == rear) {

printf("the deQueue element %d", queue[front]);

front = rear = -1;

} else {

printf("the deQueue element %d", queue[front]);

front++;

}

}

void display() {

int i;

if (front == -1 && rear == -1) {

printf("queue is empty");

} else {

for (i = front; i < rear + 1; i++) {

printf("%d", queue[i]);

}

}

}

void peek() {

if (front == -1 && rear == -1) {

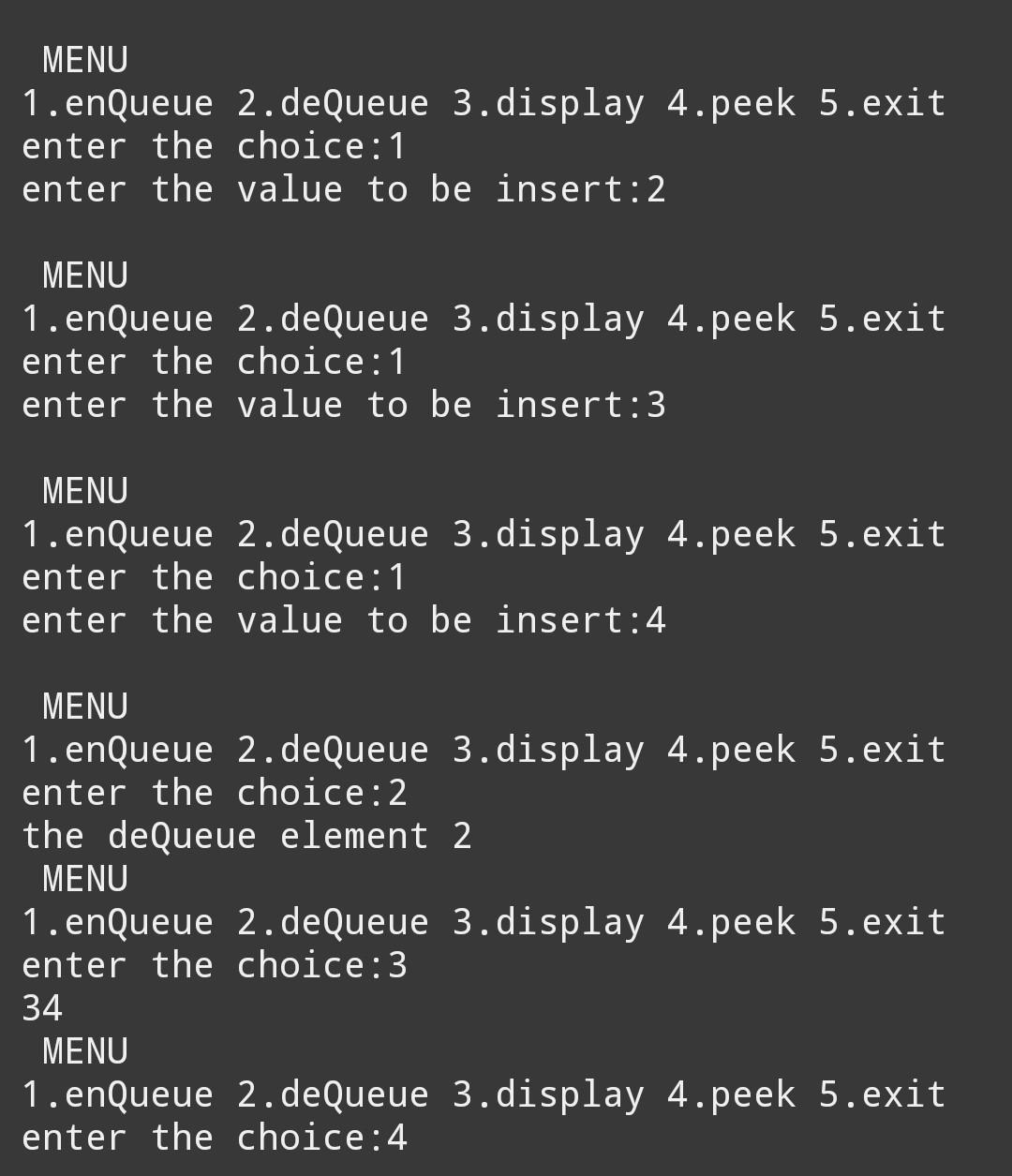
printf("queue is empty");

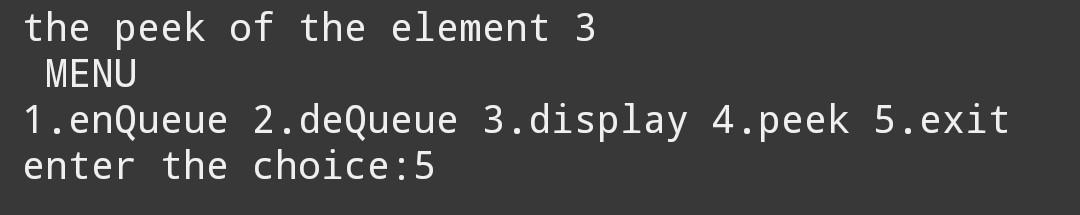
} else {

printf("the peek of the element %d", queue[front]);

}

}





#include<stdio.h>

#define SIZE 5

void enQueue(int value);

void deQueue();

void display();

void peek();

int queue[10], front = -1, rear = -1;

int value, choice;

int main() {

while (1) {

printf(" MENU\n");

printf("1.enQueue 2.deQueue 3.display 4.peek 5.exit");

printf("\nenter the choice:");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("enter the value to be insert:");

scanf("%d", &value);

enQueue(value);

break;

case 2:

deQueue();

break;

case 3:

display();

break;

case 4:

peek();

break;

case 5:

exit(0);

default:

printf("enter the valid choice");

}

}

}

void enQueue(int value) {

if (front == -1 && rear == -1) {

front = rear = 0;

queue[rear] = value;

} else {

if ((rear+1)%SIZE == front) {

printf("the queue is full");

} else {

rear = (rear+1)%SIZE;

queue[rear] = value;

}

}

}

void deQueue() {

if (front == -1 && rear == -1) {

printf("queue is empty");

} else {

if (front == rear) {

printf("the deQueue element %d", queue[front]);

front = rear = -1;

} else {

printf("the deQueue element %d", queue[front]);

front = (front+1)%SIZE;

}

}

}

void display() {

int i;

if (front == -1 && rear == -1) {

printf("queue is empty");

} else {

printf("queue: ");

i = front;

while (i != rear) {

printf("%d ", queue[i]);

i = (i+1)%SIZE;

}

printf("%d", queue[rear]);

}

}

void peek() {

if (front == -1 && rear == -1) {

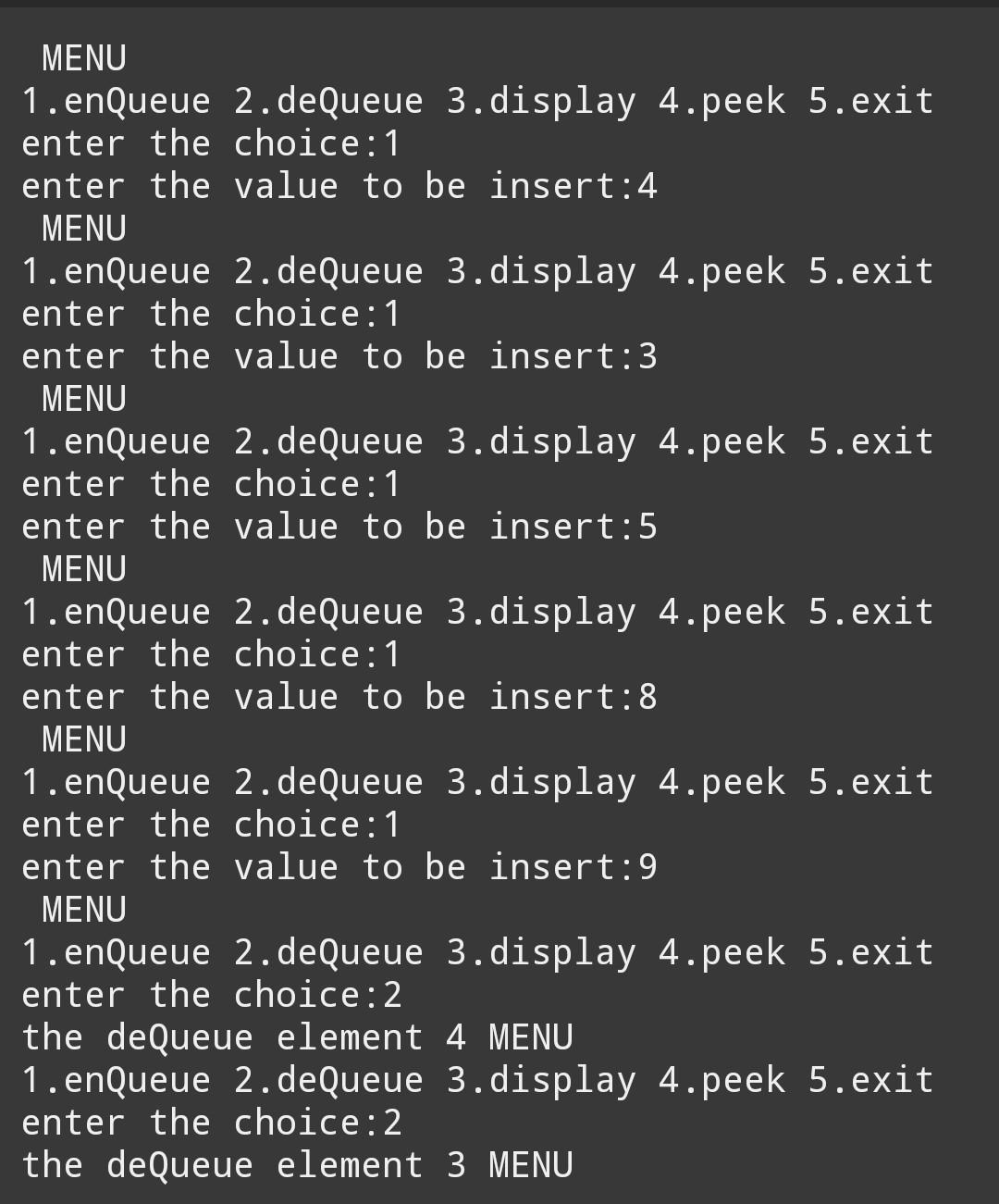
printf("queue is empty");

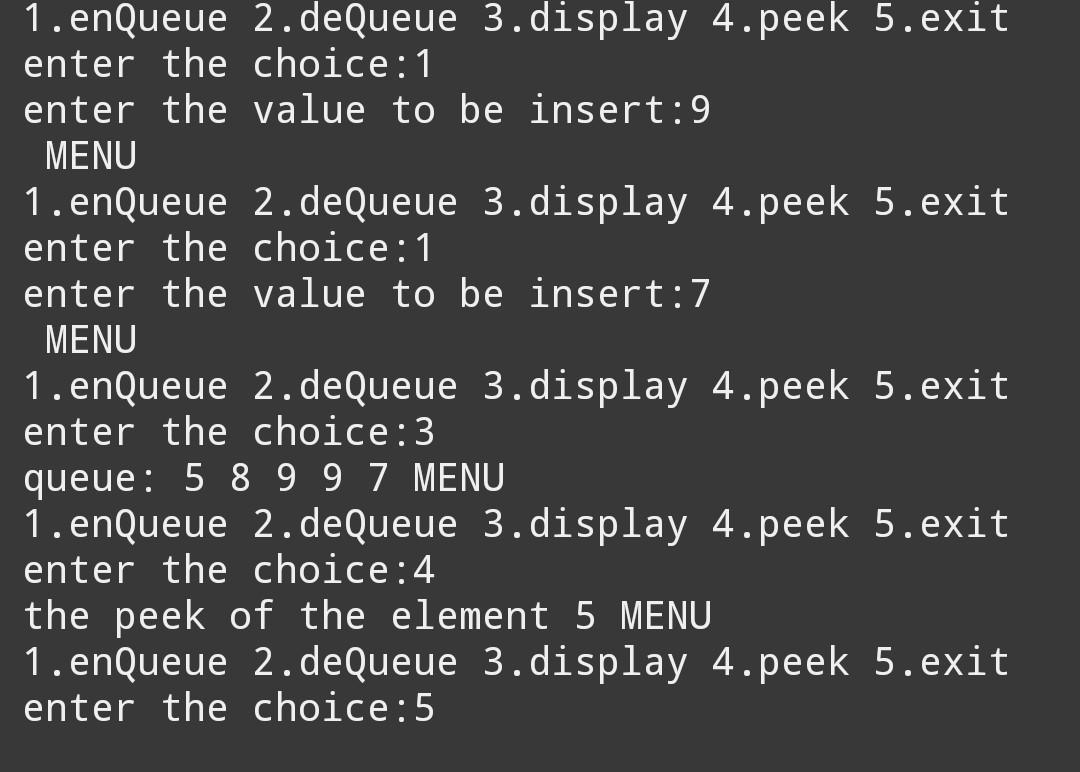
} else {

printf("the peek of the element %d", queue[front]);

}

}





#include <stdio.h>

#define N 4

int board[N][N] = {0};

void printSolution() {

for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) {

printf("%d ", board[i][j]);

}

printf("\n");

}

}

int isSafe(int row, int col) {

int i, j;

for (i = 0; i < col; i++)

if (board[row][i])

return 0;

for (i = row, j = col; i >= 0 && j >= 0; i--, j--)

if (board[i][j])

return 0;

for (i = row, j = col; j >= 0 && i < N; i++, j--)

if (board[i][j])

return 0;

return 1;

}

int solveNQUtil(int col) {

if (col >= N)

return 1;

for (int i = 0; i < N; i++) {

if (isSafe(i, col)) {

board[i][col] = 1;

if (solveNQUtil(col + 1))

return 1;

board[i][col] = 0;

}

}

return 0;

}

int solveNQ() {

if (solveNQUtil(0) == 0) {

printf("Solution does not exist");

return 0;

}

printSolution();

return 1;

}

int main() {

solveNQ();

return 0;

}

