QUEUE OPERATIONS

#include <stdio.h>

#include <stdlib.h>

#include <limits.h>

#define CAPACITY 100

typedef struct node

{

int data;

struct node \* next;

} Queue;

unsigned int size = 0;

int enqueue(Queue \*\* rear, Queue \*\* front, int data);

int dequeue(Queue \*\* front);

int getRear(Queue \* rear);

int getFront(Queue \* front);

int isEmpty();

int isFull();

int main()

{

int ch, data;

Queue \*rear, \*front;

rear = NULL;

front = NULL;

while (1)

{

printf("--------------------------------------------\n");

printf(" QUEUE LINKED LIST IMPLEMENTATION PROGRAM \n");

printf("--------------------------------------------\n");

printf("1. Enqueue\n");

printf("2. Dequeue\n");

printf("3. Size\n");

printf("4. Get Rear\n");

printf("5. Get Front\n");

printf("0. Exit\n");

printf("--------------------------------------------\n");

printf("Select an option: ");

scanf("%d", &ch);

switch (ch)

{

case 1:

printf("\nEnter data to enqueue: ");

scanf("%d", &data);

if (enqueue(&rear, &front, data))

printf("Element added to queue.");

else

printf("Queue is full.");

break;

case 2:

data = dequeue(&front);

if (data == INT\_MIN)

printf("Queue is empty.");

else

printf("Data => %d", data);

break;

case 3:

if (isEmpty())

printf("Queue is empty.");

else

printf("Queue size => %d", size);

break;

case 4:

data = getRear(rear);

if (data == INT\_MIN)

printf("Queue is empty.");

else

printf("Rear => %d", data);

break;

case 5:

data = getFront(front);

if (data == INT\_MIN)

printf("Queue is empty.");

else

printf("Front => %d", data);

break;

case 0:

printf("Exiting from app.\n");

exit(0);

default:

printf("Invalid choice, please input number between (0-5).");

break;

}

printf("\n\n");

}

}

int enqueue(Queue \*\* rear, Queue \*\* front, int data)

{

Queue \* newNode = NULL;

if (isFull())

{

return 0;

}

newNode = (Queue \*) malloc (sizeof(Queue));

newNode->data = data;

newNode->next = NULL;

if ( (\*rear) )

{

(\*rear)->next = newNode;

}

\*rear = newNode;

if ( !( \*front) )

{

\*front = \*rear;

}

size++;

return 1;

}

int dequeue(Queue \*\* front)

{

Queue \*toDequque = NULL;

int data = INT\_MIN;

if (isEmpty())

{

return INT\_MIN;

}

toDequque = \*front;

data = toDequque->data;

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

size--;

free(toDequque);

return data;

}

int getRear(Queue \* rear)

{

.

return (isEmpty())

? INT\_MIN

: rear->data;

}

int getFront(Queue \* front)

{

return (isEmpty())

? INT\_MIN

: front->data;

}

int isEmpty()

{

return (size <= 0);

}

int isFull()

{

return (size > CAPACITY);

}

OUTPUT:

