1.testqueue.c

```
#include <stdio.h>

#include "queue.h"

#include <string.h>

int main() {

    queue q;

    data d;

    qinit(&q);

    while (1) {

        printf("Enter name (or 'exit' to finish): ");

        // Read name

        scanf("%[^\n]s", d.name);

        getchar();

        // Check if the user typed "exit"

        if (strcmp(d.name, "exit") == 0) {

            break; // Exit if the user types 'exit'

        }

        // Prompt for age

        printf("Enter age: ");

        scanf("%u", &(d.age));

        getchar();

        // Enqueue only if the queue is not full

        if (!qfull(&q)) {

            enq(&q, d);

        } else {

            printf("Queue is full, cannot enqueue more data.\n");

        }

    }
    if (qempty(&q)) {
        printf("Queue is empty, no contents to display.\n");
    } else {
        // Display the contents of the queue
        printf("\nQueue contents:\n");
        while (!qempty(&q)) {
            d = deq(&q);
            printf("%s %u\n", d.name, d.age);
        }
    }
    return 0;
}
```

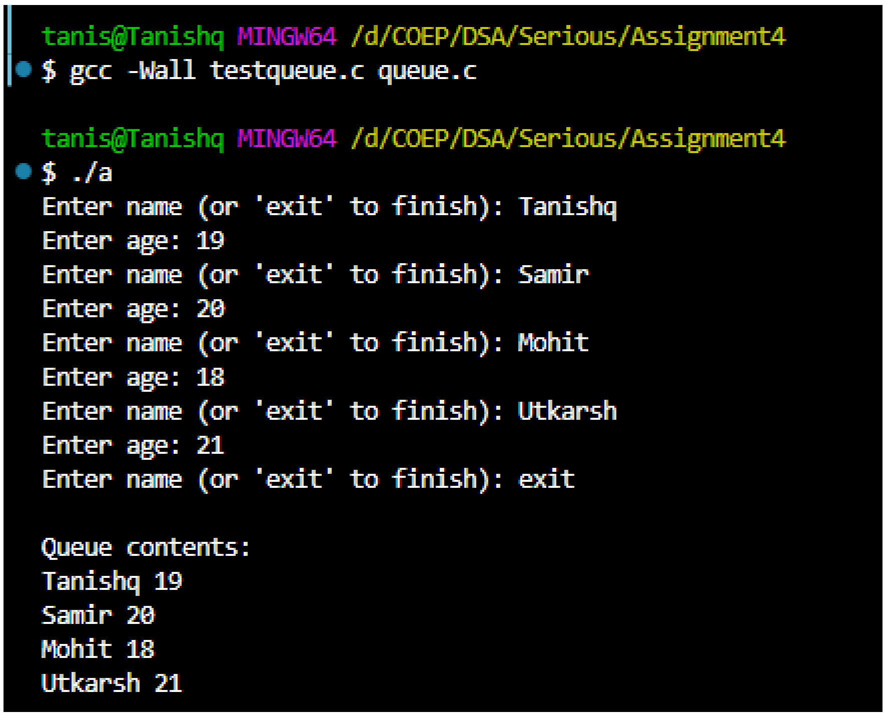
2.queue.h

```
// Data structure to hold the information
typedef struct data {
   char name[16];
   unsigned int age;
} data;

// Node structure for the queue
typedef struct node {
   data value;
   struct node *next;
} node;

// Queue structure
typedef struct {
   node *head;
   node *tail;
} queue;

void qinit(queue *q);
int qfull(queue *q);
int qempty(queue *q);
void enq(queue *q, data d);
data deq(queue *q);
```

3.queue.c

```
#include <stdio.h>

#include <stdlib.h>

#include "queue.h"

// Function to initialize the queue

void qinit(queue *q) {

    q->head = NULL;

    q->tail = NULL;

}

// Function to check if the queue is empty

int qempty(queue *q) {

    return (q->head == NULL);

}

// Function to check if the queue is full

int qfull(queue *q) {

    node *temp = (node *)malloc(sizeof(node));

    if (temp == NULL) {

        return 1; // Queue is full (memory allocation failed)

    }

    free(temp);

    return 0; // Queue is not full

}

// Function to add an element to the queue

void enq(queue *q, data d) {

    node *newNode = (node *)malloc(sizeof(node));

    if (newNode == NULL) {

        printf("Memory allocation failed, cannot enqueue data.\n");

        return;

    }

    newNode->value = d;

    newNode->next = NULL;
    if (qempty(q)) {
        q->head = newNode;
        q->tail = newNode;
        newNode->next = newNode; // Circular connection to itself
    } else {
        newNode->next = q->head; // Link new node to head
        q->tail->next = newNode; // Tail's next points to new node
        q->tail = newNode;       // Update tail to new node
    }
}
// Function to remove an element from the queue
data deq(queue *q) {
    data d;
    node *temp;
    if (qempty(q)) {
        printf("Queue is empty, cannot dequeue data.\n");
        return d; // Return an empty data struct
    }
    temp = q->head;
    d = temp->value;
    if (q->head == q->tail) { // Only one node in the queue
        q->head = NULL;
        q->tail = NULL;
    } else {
        q->head = q->head->next;
        q->tail->next = q->head; // Maintain circular connection
    }
    free(temp);
    return d;
}
```

Output:



```
tanis@Tanishq MINGW64 /d/COEP/DSA/Serious/Assignment4
$ gcc -Wall testqueue.c queue.c

tanis@Tanishq MINGW64 /d/COEP/DSA/Serious/Assignment4
$ ./a
Enter name (or 'exit' to finish): Tanishq
Enter age: 19
Enter name (or 'exit' to finish): Samir
Enter age: 20
Enter name (or 'exit' to finish): Mohit
Enter age: 18
Enter name (or 'exit' to finish): Utkarsh
Enter age: 21
Enter name (or 'exit' to finish): exit

Queue contents:
Tanishq 19
Samir 20
Mohit 18
Utkarsh 21
```