queue.h

```
#ifndef QUEUE_H_INCLUDED
#define QUEUE_H_INCLUDED
#include <iostream>

using namespace std;

typedef int infotype;

struct Queue{
   infotype info[5];
   int head;
   int tail;
};

void createQueue(Queue &Q);

bool isEmptyQueue(Queue Q);
bool isEmptyAlt2(Queue Q);
bool isEmptyAlt3(Queue Q); //belum paham

bool isFullQueue(Queue Q);
bool isFullAlt2(Queue Q);
bool isFullAlt3(Queue Q); //belum paham

void enqueue(Queue &Q, infotype X);
void enqueueAlt2(Queue &Q, infotype X);
void enqueueAlt3(Queue &Q, infotype X); //belum paham

infotype dequeue(Queue &Q);
```

```
infotype dequeueAlt2(Queue &Q);
infotype dequeueAlt3(Queue &Q); //belum paham

void printInfo(Queue Q);
#endif // QUEUE_H_INCLUDED
```

queue.cpp

```
#include "queue.h"

void createQueue(Queue &Q){
   Q.head = -1;
   Q.tail = -1;
}

bool isEmptyQueue(Queue Q){
   if(Q.head == -1 && Q.tail == -1){
      return true;
   } else {
      return false;
   }
}

bool isEmptyAlt2(Queue Q)
{
   if(Q.head == -1 && Q.tail == -1)
   {
      return true;
   } else
   {
      return false;
   }
}

bool isEmptyAlt3(Queue Q) // belum paham
{
   if(Q.head == -1 && Q.tail == -1)
```

```
    {
        return true;
    } else
    {
        return false;
    }
}

bool isFullQueue(Queue Q){
    if(Q.head == 0 && Q.tail == 4){
        return true;
    } else {
        return false;
    }
}

bool isFullAlt2(Queue Q)
{
    if(Q.head == 0 && Q.tail == 4)
    {
        return true;
    } else
    {
        return false;
    }
}

bool isFullAlt3(Queue Q) // belum paham
{
    if((Q.head == 0 && Q.tail == 4) || ((Q.head = (Q.tail + 1))))
    {
```

```
        return true;
    } else
    {
        return false;
    }
}

void enqueue(Queue &Q, infotype X){
    if(isEmptyQueue(Q) == true){
        Q.head = 0;
        Q.tail = 0;
        Q.info[Q.tail] = X;
    } else {
        if(isFullQueue(Q) != true){
            Q.tail = Q.tail + 1;
            Q.info[Q.tail] = X;
        } else {
            cout << "full queue" << endl;
        }
    }
}

void enqueueAlt2(Queue &Q, infotype X)
{
    if(isEmptyQueue(Q) == true){
        Q.head = 0;
        Q.tail = 0;
        Q.info[Q.tail] = X;
    } else {
        if(isFullQueue(Q) != true){
            Q.tail = Q.tail + 1;
```

```
        Q.info[Q.tail] = X;
      } else {
        cout << "full queue" << endl;
      }
  }

//  int i,j;
//  if(isFullAlt2(Q))
//  {
//    cout << "full queue" << endl;
//  } else if(isEmptyAlt2(Q) == true)
//  {
//    Q.head = Q.head + 1;
//    Q.tail = Q.tail + 1;
//    Q.info[Q.tail] = X;
//  }
//  } else if(Q.tail == 4)
//  {
//    i = Q.head;
//    j = 0;
//    while(i < Q.tail)
//    {
//      Q.info[j] = Q.info[i];
//      i = i + 1;
//      j = j + 1;
//    }
//    Q.head = 0;
//    Q.tail = j;
//    Q.info[Q.tail] = X;
//  } else
//  {
```

```
//       Q.tail = Q.tail + 1;
//       Q.info[Q.tail] = X;
//    }
}


void enqueueAlt3(Queue &Q, infotype X) // belum paham
{
   if(isFullAlt3(Q))
   {
      cout << "full queue" << endl;
   } else if(isEmptyAlt3(Q) == true)
   {
      Q.head = Q.head + 1;
      Q.tail = Q.tail + 1;
      if (Q.tail == 4)
      {
         Q.tail = 0;
      } else
      {
         Q.tail = Q.tail + 1;
      }
      Q.info[Q.tail] = X;
   }
}

infotype dequeue(Queue &Q){
   infotype X;
   if(Q.tail == 0){
      X = Q.info[0];
      Q.head = -1;
```

```
        Q.tail = -1;
    } else if (Q.tail != 0){
        int i, y, X;
        y = Q.head;
        X = Q.info[y];
        for(i=Q.head;i<=Q.tail;i++){
            Q.info[i] = Q.info[i+1];
        }
        Q.tail = Q.tail - 1;
    } else {
        cout << "Queue kosong" << endl;
    }
    return X;
}

infotype dequeueAlt2(Queue &Q)
{
    infotype X;
    if(Q.tail == Q.head)
    {
        X = Q.info[Q.head];
        Q.head = -1;
        Q.tail = -1;
    } else if(Q.head != Q.tail)
    {
        X = Q.info[Q.head];
        Q.head = Q.head + 1;
    } else
    {
        cout << "Queue Kosong" << endl;
    }
```

```
   return X;

//    if(isEmptyAlt2(Q))
//    {
//        cout << "Stack Kosong" << endl;
//    } else
//    {
//        X = Q.info[Q.head];
//        if(Q.head == Q.tail){
//            Q.head = -1;
//            Q.tail = -1;
//        } else
//        {
//            Q.head = Q.head + 1;
//        }
//        return X;
//    }
}

infotype dequeueAlt3(Queue &Q) //belum paham
{
   infotype X;
   if(isEmptyAlt3(Q))
   {
      cout << "Stack Kosong" << endl;
   } else
   {
      if(Q.tail == Q.head)
      {
         X = Q.info[Q.head];
         Q.head = -1;
```

```
            Q.tail = -1;
        } else if(Q.head = 4 && Q.tail != Q.head)
        {
            X = Q.info[Q.head];
            Q.head = 0;
        } else {
            X = Q.info[Q.head];
            Q.head = Q.head + 1;
        }
    }
    return X;
}

void printInfo(Queue Q){
    if(Q.head == -1 && Q.tail == -1){
        cout << " " << Q.head << " - " << Q.tail << " | " << "empty queue" << endl;
    } else {
        int i;
        for (i = Q.head; i <= Q.tail; i++){
        }
        cout << " " << Q.head << " - " << Q.tail << "   | ";

        for (i = Q.head; i <= Q.tail; i++){
            cout << Q.info[i] << " ";
        }
        cout << endl;
    }
}
```

main.cpp

```
#include "queue.h"


int main()
{
    cout << "Hello world!" << endl;
    cout << "----------------------" << endl;
    cout << " H - T   | Queue Info " << endl;
    cout << "----------------------" << endl;
    Queue Q;
    infotype X;
    createQueue(Q);
    printInfo(Q);
    enqueue(Q, 5); printInfo(Q);
    enqueue(Q, 2); printInfo(Q);
    enqueue(Q, 7); printInfo(Q);
    X = dequeue(Q); printInfo(Q);
    X = dequeue(Q); printInfo(Q);
    enqueue(Q, 4); printInfo(Q);
    X = dequeue(Q); printInfo(Q);
    X = dequeue(Q); printInfo(Q);


    cout << endl << endl;


    cout << "Hello world!" << endl;
    cout << "----------------------" << endl;
    cout << " H - T   | Queue Info " << endl;
    cout << "----------------------" << endl;
    createQueue(Q);
    printInfo(Q);
```

```
    enqueueAlt2(Q, 5); printInfo(Q);
    enqueueAlt2(Q, 2); printInfo(Q);
    enqueueAlt2(Q, 7); printInfo(Q);
    X = dequeueAlt2(Q); printInfo(Q);
    X = dequeueAlt2(Q); printInfo(Q);
    enqueueAlt2(Q, 4); printInfo(Q);
    X = dequeueAlt2(Q); printInfo(Q);
    X = dequeueAlt2(Q); printInfo(Q);

    cout << endl << endl;

    cout << "Hello world!" << endl;
    cout << "----------------------" << endl;
    cout << " H - T   | Queue Info " << endl;
    cout << "----------------------" << endl;
    createQueue(Q);
    printInfo(Q);
    enqueueAlt3(Q, 5); printInfo(Q);
    enqueueAlt3(Q, 2); printInfo(Q);
    enqueueAlt3(Q, 7); printInfo(Q);
    X = dequeueAlt3(Q); printInfo(Q);
    X = dequeueAlt3(Q); printInfo(Q);
    enqueueAlt3(Q, 4); printInfo(Q);
    X = dequeueAlt3(Q); printInfo(Q);
    X = dequeueAlt3(Q); printInfo(Q);

    return 0;
}
```