# Inlämningsuppgift 1

## Inledning:

Denna uppgift belyser konceptet **ADT (Abstract Data Type)** och specifikt de abstrakta datatyperna **Lista** och **Kö**.

I denna uppgift kommer du att implementera en kö samt en lista vilken har ett cirkulärt beteende. Listan och kön ska användas i en variant av Josephus problem. Kod för Josephus problem tillhandahålls.

Implementationsdetaljer:

- Template
- Inre klass
- Pekare
- Abstrakt klass
- Rent virtuella funktioner

På kursens hemsida finner du följande filer vilka ska ingå i din lösning:

- ICircularDoubleDirectedList.h
- IQueue.h
- Josephs.cpp

Du finner även följande filer vilka du ska använda för att testa din implementation:

- TestFunctionalityOfList.cpp
- TestDeepCopyingOfList.cpp
- TestFunctionalityOfQueue.cpp
- TestDeepCopyingOfQueue.cpp

**Krav på uppbyggnad:**

Listan ska som datastruktur länka noder cirkulärt i två riktningar. En inre klass `Node` ska definieras.

Gränssnittet för listan ges av klassmallen `ICircularDoubleDirectedList` enligt:

```
enum direction{FORWARD, BACKWARD};
template <typename T>
class ICircularDoubleDirectedList
{
public:

	virtual ~ICircularDoubleDirectedList() {};
	virtual void addAtCurrent(const T& element) = 0;
	virtual T getElementAtCurrent() const = 0;
	virtual void removeAtCurrent() = 0;
	virtual int size() const = 0;
	virtual void changeDirection() = 0;
	virtual void moveCurrent() = 0;
	virtual direction getCurrentDirection() const = 0;
};
```

Gränssnittet för kön ges av `IQueue` enligt:

```
template <typename T>
class IQueue
{
public:
	virtual ~IQueue() {};
	virtual void enqueue(const T& element) = 0;
	virtual void dequeue() = 0;
	virtual T front() const = 0;
	virtual bool isEmpty() const = 0;
};
```

Datastruktur kan för kön vara antingen enkellänkning eller en array. Observera att kön är obegränsad avseende antalet element som kan ingå i kön.

**Del I:**

Klassmallen `ICircularDoubleDirectedList` finns i filen ICircularDoubleDirectedList.h. Denna ska ingå i din lösning, d.v.s i det Visual Studio projekt som kommer att innehålla filerna för lösningen till uppgiften.

Du ska dessutom skapa en klassmall benämnd `CircularDoubleDirectedList` vilken ärver publikt från `ICircularDoubleDirectedList`.

I klassmallen `CircularDoubleDirectedList` definieras klassen för dubbelriktade noder. Dubbelriktade noder länkas för att erhålla en dubbellänkad cirkulär lista. Dessutom deklareras följande medlemsvariabler:

- `current` vilken pekar ut aktuell nod
- `nrOfElements` vilken innehåller antalet element i listan
- `currentDirection` vilken innehåller aktuell riktning

När en lista skapas ska dess aktuella riktning vara FORWARD.

Beskrivning av medlemsfunktioner:

- `addAtCurrent(…)`
  - Om listan är tom skapas en nod för `element` vilken blir den enda noden i listan. Den nya noden är därefter aktuell nod.
  - En ny nod skapas för `element` vilken placeras före aktuell nod om riktningen är BACKWARD och efter aktuell nod om riktningen är FORWARD. Den nya noden ska därefter bli aktuell nod.
- `getElementAtCurrent()`
  - Om listan är tom kastas strängen ” Exception: call of getElementAtCurrent on empty list”som undantag.
  - Returnerar det element som är aktuellt.
- `removeAtCurrent()`
  - Om listan är tom kastas strängen ”Exception: call of removeAtCurrent on empty list” som undantag.
  - Aktuell nod tas bort. Om listan endast har en nod blir listan tom och annars blir aktuell nod den nod som är före om riktningen är BACKWARD och den nod som är efter om riktningen är FORWARD.
- `size()`
  - Returnerar antalet element som finns i listan.
- `changeDirection()`
  - Ändrar riktningen för listan.

- `moveCurrent()`
  - Om listan är tom kastas strängen ” Exception: call of moveCurrent on empty list”som undantag.
  - Ändrar `current` till ”nästa” nod enligt den riktning som ges av `currentDirection`.
- `getCurrentDirection()`
  - Returnerar listans aktuella riktning.

Klassmallen `IQueue` finns i filen IQueue.h. Denna ska ingå i din lösning, d.v.s i det Visual Studio projekt som kommer att innehålla filerna för lösningen till uppgiften.

Du ska dessutom skapa en klassmall benämnd `Queue` vilken ärver publikt från `IQueue`. (Du kan använd den implementation du gjort i samband med första laborationstillfället.)

Medlemsfunktionerna `dequeue()` resp `front()` ska, om kön är tom, kasta strängen ”Exception: empty queue” som undantag.

**Testning:**

Genomför tester för listan och kön med följande testprogram:

- TestFunctionalityOfList.cpp
- TestDeepCopyingOfList.cpp
- TestFunctionalityOfQueue.cpp
- TestDeepCopyingOfQueue.cpp

Testningen ska genomföras i debug-läge för att upptäcka eventuella minnesläckor. Vid exekvering anges förväntade utskrifter och därefter den utskrift som ges av programmet när din lista och din kö används. Dessa utskrifter ska vara desamma.

Inga minnesläckor får förekomma.

## Del II:

Listan och kön ska nu användas för att hantera en del i implementationen för en speciell variant av *Josephus problem*. Du behöver tillföra din implementation av klassmallarna `CircularDoubleDirectedList.h` och `Queue.h`.

### Beskrivning av Josephus problem:

Ett antal personer som står i en cirkel ska räknas ut (som t ex görs i många räkneramsor) en efter en. Beräkningen av vem som ska räknas ut härnäst görs genom att man räknar ett antal från en viss position i cirkeln. (Josephus problemet beskrivs vanligen betydligt blodigare än så.)

I denna version räknar man framåt varannan gång och bakåt varannan gång. Till slut kommer det bara att finnas en person kvar och denna person är den som vunnit.

Anta exempelvis att personerna A, B, C, D och E står i en cirkel och att beräkning börjar på A samt att det antal man beräknar är 8.

Riktningen är från början framåt, d.v.s mot B. Första personen som kommer att räknas ut blir då C (räknat cirkulärt framåt från A ger följden: A->B->C->D->E->A->B->C). C lämnar cirkeln och därefter är startpositionen D. Riktningen ändras dessutom till bakåt d.v.s mot B. Andra personen som kommer att räknas ut är E (D->B->A->E->D->B->A->E). Startpositionen är nu D och riktningen ändras återigen till framåt. Denna hantering fortgår tills det endast finns en person kvar vilket kommer att vara personen B. Personerna kommer i just detta exempel att räknas ut i ordningen C, E, A, D och B är den som står som vinnare.

Om personerna är A, B, C, D, E, F, G och räknaren är 5 blir uträkningsordningen: E, A, F, G, B, C och den som står som vinnare är D.

### Slutprodukt och inlämning:

Slutprodukten är ett fungerande program som motsvarar beskrivningen av Josephus problem ovan där dina filer (listan och kön) ingår tillsammans med de filer som tillhandahållits. Dina implementationer av listan och kön ska vara testade med de testprogram som tillhandahållits och inte ge några minnesläckor. Du lämnar in alla de filer som motsvarar slutprodukten och dessutom testprogrammen..