Labb 6: SharedPtr

Ni ska implementera er egen variant av std::shared_ptr och std::weak_ptr, nedan kallade SharedPtr och WeakPtr. Den ska även vara exception safe (vilket inte gör stor skillnad).

Observera att ni skall använda er av minnesdump så ni ser att det inte blir någon minnesläcka. Se tipsen på ITs.

G nedan står för vad som krävs för G och VG för VG. WeakPtr behövs för VG.

Operationerna nedan stämmer med STLs förutom att STL har fler member functions och static functions.

	shared_ptr	**weak_ptr**	**Kommentar**
- Konstruktor som tar:			
o void (inget)	G	VG	
o nullptr	G		
o En pekare	G		
o En SharedPtr&	G	VG	
o En SharedPtr&&	G		Move-constructor
o En WeakPtr	VG	VG	Throw exception!
- Destruktor	G	VG	
- Tilldelning från en			
o En SharedPtr&	G	VG	
o En SharedPtr&&	VG		Move-assignmnet
o En WeakPtr		VG	
- Jämförelse med (== och <)			
o nullptr	G		
o En SharedPtr	G		Jämförelse av den underliggande pekaren
- operator*	G		
- operator->	G		
- operator bool	G		True om det finns ett objekt
- funktioner:			
o reset(T*=nullptr)	G		Byter objekt
o get()	G		Ger tillgång till pekaren
o unique()	G		
o lock()		VG	
o expired()		VG	

För VG så krävs även.

- så fort en weakPtr används så ska den om den är expired räkna ner referensräknaren så att referensräknarobjektet kan deletas så fort som möjligt.
- Fixa konstruktorerna så att de inte bara kan ta en pekare av samma typ utan vilken kompatibel pekare som helst.

Testprogrammet i Main.cpp

Observera att det testprogram som finns i Main.cpp bara är en hjälp och varken fullständigt eller garanterat helt korrekt. Det är möjligt att testprogrammet kör felfritt fast er lösning är felaktig. Det är även möjligt – men inte troligt – att er lösning är korrekt fast testprogrammet inte kör/kompilerar felfritt.