# Laborationsuppgift 3 – Nyheter i C++

Laborationsrapporten skall lämnas in på papper och innehålla följande:

Ett försättsblad med:

- Kursens namn och termin
- Laborationens nummer
- Studentens namn och användar-id
- Handledarens namn (Erik Eliasson i detta fall)

För varje uppgift en kort beskrivning av vad uppgiften är (med egna ord, inte direkt kopierat från uppgiften).

Källkod (välkommenterad).

Utskrifter från testkörningar, så att vi ser att programmet fungerar som tänkt. Beskriv även de begränsningar som din lösning har.

Avslutningsvis några ord om eventuella problem du stött på och andra funderingar du kan ha haft.

Uppgiften skall lösas enskilt. Naturligtvis får ni hjälpa varandra, men ni skall skriva programmen och rapporten var för sig, och ni skall också muntligt kunna förklara vad programmen gör.

## Lösningen skall vara inlämnad senast klockan 12.00 på Tisdagen den 23/5.

## Del 1, Templates

Deluppgift ett är en övning i att använda templates i C++. Du ska skriva en klass för en mängd nycklar där varje nyckel har ett värde associerat till sig. Typen för nyckeln och värdet anges när man skapar objektet.

Klassen ska innehålla funktionerna

- put – lägger till en nyckel med motsvarande värde till mängden
- get – returnerar värdet som är associerat till en nyckel
- contains – returnerar true om en given nyckel finns i mängden och false annars
- is_full – returnerar true om mängden är full och inga fler nycklar kan läggas till

Klassdeklarationen kan se ut som nedan

```
template<class Key_Type, class Item_Type, const int MAX_ELEMENTS=10>
class Association_Set{
public:
   Association_Set();
   ~Association_Set();
   void put(Key_Type key, Item_Type item);
   Item_Type get(Key_Type key);
   bool contains_key(Key_Type key);
   bool is_full();
private:
   int x;
   Key_Type keys[MAX_ELEMENTS];
   Item_Type items[MAX_ELEMENTS];
   int pos;
};
```

(default-parametrar i templates är ett relativt nytt tillägg, så om det inte fungerar med din kompilator får du hitta på en egen lösning)

Ett enkelt exempel på hur man kan använda klassen är

```
int main(){
   Association_Set<string, long, 100> birthdays;
   birthdays.put("Erik", 770424);
   birthdays.put("Therese", 790210);
   cout << "Eriks birthday is " << birthdays.get("Erik") << endl;

   Association_Set<char, string> animals;
   animals.put('G', "Giraffe");
   animals.put('E', "Elephant");
   animals.put('P', "Penguin");

   cout << "An animal stating with the letter E is " <<
         animals.get('E') << " and one stating with P is " <<
         animals.get('P') << endl;
   return 0;
}
```

Fel som att användaren lägger till nycklar när mängden är full eller att man lägger till en nyckel som redan finns kan hanteras med hjälp av undantag (exceptions).

## Del 2, Algoritmbiblioteket

**2.1**. Vektor-funktioner ur Algoritmbiblioteket
Skapa en vector utifrån en vanlig array mha back_inserter(), sortera den med sort() och skriv ut den med copy() och utskriftsiteratorn std::ostream_iterator<int>(std::cout, ",")

**2.2**. Undantagshantering
Skriv ett litet program som illustrerar C++ exceptions. Använd vector-klassens at() funktion. Vad ger what() för meddelande?

**2.3** Iteratorer
Illustrera användningen av en iterator med ett lämpligt programexempel!

Lycka till!