Laboration 1: Pekare

```
//Print the array
cout << "This is the array containing the random numbers:\n";
for(int *i=numbers; i != numbers + arrLength; i++) {
    if((*i % 200) == 0 && *i > 200)
        cout << "\n";
    else
        cout << *i << ' ';

    // Get statistics
    // In the continuation of getting, lowest, largest then adding to sum.
    if(*i < lowest)
        lowest =  *i;
    if(*i > largest)
        largest = *i;
    sum += *i;
}

cout << "\nThe min, max and sum value is: " << lowest << ", " << largest << ", " << s


delete []numbers;
return 0;
```

Utrustning:

Jag har använt mjukvaran Eclipse med plugin för utveckling av C/C++ programvaror. Hela utvecklingen och miljön kördes under Mac OSX Snow Leopard.

Beskrivning av uppgiften:

Jag får börja med att ursäkta den sena inlämningen. Jag överskattade nog lite min kunskap i C gällande specifikt pekare och hur snabbt jag skulle lösa uppgiften korrekt.

Jag hade också lite problem gällande kompileringen, jag var tvungen att köpa och ladda ner xcode till Mac för att få miljön att fungera som bäst (rekommendation från eclipse hjälpsidor). Med tillfälligt långsam internetuppkoppling och ett stort program så tog det en bra stund innan jag fick ner hela programmet och kunde kompilera min kod. Det jag kom på i efterhand var att jag kunde ladda ner och installera gcc självständigt då jag tror att det är den kompilatorn man får med i xcode tools.

Hur som helst var det mitt fel att jag inte började i god tid, vilket jag ursäktar mig för.

Uppgiften gick ut på att skriva ett program som skapade en vektor dynamiskt och fylla den med slumpmässiga värden. Sedan så skall man referera till vektorn med pekare och föra lite statistik kring vektorn. I slutändan skulle man skriva ut allt till skärmen så att användaren kunde se resultatet och statistiken.

Problem man kan få är nog främst i de specifika kraven runt pekare och de slumpmässiga talen. Det kan nog vara svårt för en del att förstå hur man skapar slumpmässigt negativa tal.

Lösningen av uppgiften:

I mitt program så valde jag inte att skapa några externa funktioner. Varför jag valde det är på grund av att man inte behöver upprepa något arbete i koden. Det vill säga, att koden är extremt sekventiell och körs rakt igenom en gång och därför skulle det vara onödigt med en funktion då samma funktionalitet aldrig behövs igen.

De enda två problem jag hade med uppgiften var de jag belyste i beskrivningen av uppgiften. Då jag kommer från en lång period av java-kodande med ett väldigt stort standardbibliotek så känns det konstigt att börja med C/C++ igen. Jag är väldigt van vid java-syntax och tänk när man bygger upp program och det ska bli spännande att få se om jag får problem med det när vi dyker in djupare bland klasser.

Det första problemet jag fick var att skapa slumpmässiga värden. Inte positiva värden, det är jag mycket van vid sen den C-kursen jag gick för cirka ett år sedan. Men när det kom till negativa slumpmässiga värden så var det faktiskt något helt nytt. Det tog en stund innan den matematik jag läst tidigare återfann sig i minnet och att jag förstod att alla tal kan man multiplicera med -1 för att få dessa negativa.

Då löste jag uppgiften genom att slumpa fram positiva tal, sen slumpa fram ett tal mellan 0-1 för varje slumpmässiga tal. Om det var en etta så gjorde jag talet negativt genom att muliplicera med -1 och annars lät jag det var.

Min ursprungliga idé var att jag delade upp ”rangen” så att alla tal över 2500 gjorde jag till negativa, men det är ju knappast så slumpmässigt, så det bytte jag ut!

Mitt andra problem kom till pekare i for-loopen och jag fick krupp när jag skulle införa pekare till vektorn istället för ett heltal till att representera varven. Problemet var att jag pekade en gång för mycket genom att skriva *(array + *i). Det jag inte förstod då var att i redan pekade på array och att jag bara kunde skippa hela rasket och bara skriva *i.

Min lösningen fungerar som så att den tar input från användaren och försöker jämföra med en int. Om det inte går (det vill säga om input:en var något annat) så låter programmet bara användaren försöka igen tills användaren lyckas.

När programmet har fått en int av användaren så skapar programmet en ny array med så många platser som användaren angav. När det är klart fylls array:n med slumpade tal som kan skilja sig från -5000 till 5000.

Nästa uppgift programmet har är att skriva ut array:n med hjälp av pekare och dessutom föra statistik kring array:n. I samma loop så printar den ut varje post i array:n och jämför/sparar ner statistiken i olika variabler. I loopen så gör programmet en ny rad och hoppar över vart 200:de rad.

När loopen är slut så skrivs statistiken ut snyggt och prydligt och sedan avslutas programmet.

Urdrag ur programmet:

```
Please enter a valid number: 10
You entered: 10

This is the array containing the random numbers:
-956 1615 3182 773 4300 799 -2382 -3249 893 923
The min, max and sum value is: -3249, 4300, 5898
```

Slutsatser och kommentarer:

Mina slutsatser av denna laboration är att det ger en djupare kunskap inom hanteringen av dynamiska vektorer, pekare och slumpmässiga tal. Beroende på utformning så kan det nog också ge en djupare kunskaper om ex. ”signed” och ”unsigned” och vad det innebär.

Referenser:

Jag har främst använt mig av lektionsmaterialet och kodexemplen som är bifogade i samma dokument. Annars har jag använt mig av kodexempel och beskrivningar från cplusplus.com (http://cplusplus.com/reference/) och deras tillhörande forum.

Det kan förekomma att jag tittat på exempel från daniweb eller liknande forum/communityn (http://www.daniweb.com/) men dessvärre kommer jag inte ihåg hurdana jag verkligen har använt något därifrån då dum som jag var stängde alla tabbar i firefox innan jag började skriva detta dokument. Det allra mesta har jag bara skrivit direkt från min grundläggande kunskap och mina gamla programmeringserfarenheter i språk som Java, php, c och så vidare.