Inför tentamen

Som exempel på möjliga frågor på tentamen gicks det igenom ett par extentor från kursen C++-programmering, EDA031 som ges på LTH. Denna kurs har högre förkunskapskrav (bl. a. algoritmer och datastrukturer) och tydligare inriktning på att använda standardbiblioteken än EDAF30, så även de frågor som innehållsmässigt passar skulle antagligen förtydligats och/eller formulerats om för en tentamen på EDAF30.

Tentamen 2010–03–11

Kommentarer om hjälpmedel etc.

För EDAF30 bör andra stycket ändras till ungefär:

Du ska i dina lösningar visa att du behärskar C++ och även att du kan använda C++ standardklasser. ”C-lösningar” kan ge poängavdrag, även om de är korrekta.

Uppgift 1

Att känna igen och kunna åtgärda minnesläckor (a) och b)) är viktigt och har tagits upp på en labb (Word / Dictionary). c) är en ”lurighet” som kräver en del eftertanke. Användningen av set för att hålla en mängd sorterad och speciellt att som i d) ange en jämförare (funktor) som extra typparameter till set har inte tagits upp på kursen. (Att använda egna eller färdiga funktorer som parameter till standardalgoritmer togs däremot upp på föreläsning 10.) Att i e) skriva en överlagrad utmatningsoperator (>>) skall inte vara någon svårighet.

(Uppgift 2)

För inriktad på datastrukturer för att vara ”rättvis”.

Uppgift 3

Någon träning i att skriva program som läser parametrar från kommandoraden har inte getts på kursen. Deluppgift a) hade formulerats om – antagligen till programmet frågar interaktivt efter filnamn. Den iterativa lösningen till b) använder en stack, vilket inte har gåtts igenom på kursen i den grad som krävs här. Det finns dock en rekursiv lösning som inte använder någon explicit stack:

```
void XMLParser::parse() throw(parse_error) {
   Tag tag = get_tag();
   if (tag.kind != Tag::open) {
      throw parse_error();
   }
   parse(tag);
   if (! in.eof()) {
      throw parse_error();
   }
}

void XMLParser::parse(struct &Tag compare) throw(parse_error) {
   Tag tag = get_tag();
   while (tag.kind == Tag::open) {
      parse(tag);
      tag = get_tag();
   }
   if (compare.text != tag.text) {
      throw parse_error();
   }
}
```

Tentamen 2010–04–13

(Uppgift 1)

Att uppgiften är template-baserad ger en hel del svårigheter i lösningen. Den matematiska notationen kan också vara ett stort hinder för förståelsen av uppgiften. Kodningen av lösningen är annars ganska rättfram.

(Uppgift 2)

Förutom att uppgiften bygger på uppgift 1 så har implementationen av iteratorer knappt tagits upp under kursen (exempel finns dock i läroboken). Hade kunnat ges som separat uppgift med utförligare instruktioner om vad som behöver implementeras.

Uppgift 3

Binära träd har tagits upp på föreläsning om dynamiska datastrukturer, och användningen i uppgiften bör inte ställa till problem. Användningen av `struct` istället för `class` kan däremot vara förvirrande. Ett problem med lösningen är hur man bäst delar upp en sträng i delsträngar. Tekniken med `istringstream` har bara tagits upp ytligt på föreläsningarna. Ett alternativt sätt att dela upp i delsträngar är att använda `substr()` (ev. tillsammans med `find_first_of()` och `find_firstNot_of()`). Ytterligare en möjlighet är att gå igenom meddelandesträngen tecken för tecken – stega i trädet på punkt och streck och när man kommer till blanktecken läggs det aktuella teckenvärdet till resultatsträngen och man kan fortsätta med att avkoda nästa tecken.

(Uppgift 4)

Förutsätter en större kunskap om både om standardbiblioteken och templates än vad som kan krävas på kursen.

Sammanfattning

Någon av de uppgifter som gicks igenom (uppgift 1 och 3 från 2010–03–11 samt uppgift 3 från 2010–04–13) skulle efter omformulering kunna vara med på en tentamen på EDAF30, men skulle då antagligen räknas till ”svårare halvan”. Resterande uppgifter bör vara av mer ”rättfram” karaktär, men med växlade omfång. En av uppgifterna bör vara av typen ”hitta och rätta fel”.