Kodprov "testomgång" #4

Instruktioner

Checka ut ditt repo: `git clone http://USERNAME:PASSWORD@wrigstad.com/git/USERNAME.git`

Nu får du en katalog `USERNAME` som beroende på vilken tenta du skriver kan innehålla filer som du skall bygga vidare på. Vissa tentor har inga uppgifter med utdelad kod i katalogen `utdelad-kod`. Du skall *inte* lägga dina filer under denna katalog!

Du lämnar in tentan via git. Du kan skriva `git status` för att få se vilka filer som inte är incheckade, använda `git add` för att lägga till filer, `git commit` för att checka in tillagda filer, etc. **OBS!** Om du inte skriver `git push origin master` efter din sista incheckning finns tentan bara på din lokala hårddisk och har då *inte* lämnats in!

Skicka också hela din tenta i ett zip-arkiv till `tobias.wrigstad@gmail.com`.

Uppgifter

1. Denna uppgift går ut på att implementera ett enkelt C-bibliotek `pair` som implementerar ett par som en struct med två fält "first" och "second". Bibliotekets funktioner (med några detaljer utelämnade) finns i den utdelade filen `pair.h`.

   Skriv färdigt biblioteket (structar, funktioner, funktionshuvuden, makron, med mera), både `pair.c` och `pair.h`! Se till att paret är ordentligt *inkapslat* så att det enda sättet en klient till biblioteket kan ändra fält i ett par är via metoderna i `pair.h`. Definiera också typen `T` i `pair.h` så att par kan innehålla godtyckliga data.

   Se också till att en klient inte får problem med multipla funktionsdefinitioner vid multipel inkludering av `pair.h`.

   Att lämna in:

   (a) `pair.c`

   (b) `pair.h`

   (c) Filerna skall ligga i katalogen `uppgift1`

2. Det utdelade (ofärdiga) Java-paketet `ms` modellerar en minneshanterare som använder mark-sweep-algoritmen. Man kan allokera ett utrymme med `alloc` och också anropa `gc` för att städa bort skräp.

   – Skriv klart metoden `mark` i klassen `Chunk`.

   – Skriv klart metoden `sweep` i klassen `MemoryManager`.

   – Ändra implementationen så att minnet inte fragmenteras, d.v.s. flera fria block som ligger efter varandra i minnet skall slås samman till en enda stor chunk under sweep-fasen.

   – Ändra implementationen av `firstFreeBlock` så att om man vill allokera N bytes, hittar en chunk med M bytes, där $M > N$, så skall chunken brytas upp i två – en med storleken N och en med storleken $M - N$.

   Att lämna in:

   (a) `MemoryManager.java`

   (b) `Chunk.java`

   (c) Filerna skall ligga i katalogen `uppgift2`

Ledning Mark-sweep algoritmen traverserar heapen med start i alla chunkar (alltså allokerade block) i listan rootset. För varje chunk som kan nås vill vi sätta dess `marked`-flagga till **`true`** för att visa att chunken fortfarande går att nå från det körande programmet, samt markera alla chunkar som är direkt nåbara från den aktuella chunken (variabeln `pointers_in_chunk`).

När alla nåbara objekt har markerats vill vi gå igenom hela listan av allokerat minne och för varje chunk som inte är markerad vill vi flytta den till listan över ledigt minne.