Tentamen
C++-programmering

2011–03–08, 8.00–13.00

Hjälpmedel: En valfri C++-bok. OH-bilderna från föreläsningarna är *inte* tillåtna.

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

Uppgifterna ger preliminärt $8 + 10 + 12 + 8 + 12 = 50$ poäng. För godkänt krävs 25 poäng (3/25, 4/33, 5/42).

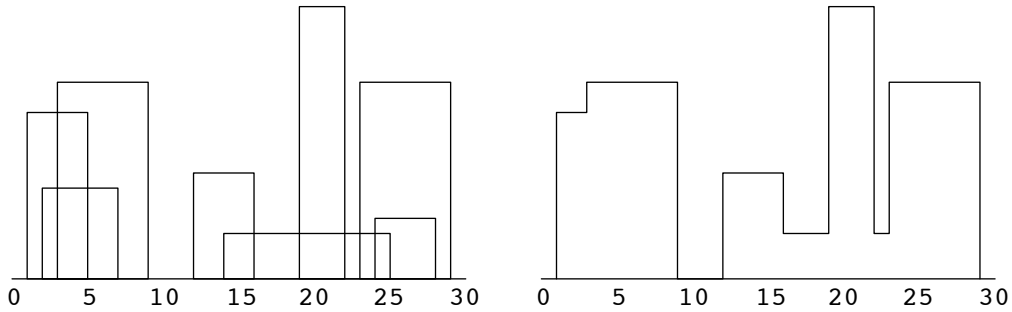
---

1. Skriv ett program som 1) läser in ord från standard input, 2) skriver ut orden på standard output, sorterade först efter längd och sedan alfabetiskt. Indata består av ord (bara små bokstäver) åtskilda av ”whitespace”.

2. En stad står på plan mark och alla byggnader är rektangulära. Vi betraktar staden rakt från sidan. En byggnad beskrivs av tre positiva heltal (`l`, `r`, `h`) där `l` och `r` är vänster och höger x-koordinat och `h` är höjden. I en viss stad beskrivs alla byggnaderna av taltripplarna:

   ```
   (1,5,11) (2,7,6) (3,9,13) (12,16,7) (14,25,3) (19,22,18) (23,29,13) (24,28,4)
   ```

   Om vi ritar upp byggnaderna ser det ut som i den vänstra bilden nedan. Om vi tittar på staden i motljus ser vi bara byggnadernas konturer mot himlen (stadens ”skyline”) som i den högra bilden.

   Konturerna i den högra bilden kan beskrivas med följande sekvens av talpar:

   ```
   1/11 3/13 9/0 12/7 16/3 19/18 22/3 23/13 29/0
   ```

   I ett talpar `m/n` är `m` x-koordinaten för en vertikal linje, `n` y-koordinaten för den horisontella linje som börjar i `m`.

   Implementera klassen `Skyline`, som beskriver konturerna:

   ```
   class Skyline {
   public:
       Skyline(); // an empty skyline
       void add(size_t l, size_t r, size_t h); // add a building with coordinates
                                               // l and r and height h
       vector<pair<size_t, size_t> > getAsSequence() const; // create the sequence
   };
   ```

---

3. Som bekant har behållarklasserna i STL både iteratorer och ”baklängesiteratorer”. För vektorer (inbyggda arrayer) fungerar pekare som iteratorer, men det finns inga baklängesiteratorer. Du ska skriva en klass `rai` (står för `reverse_array_iterator` men är kortare att skriva) vars objekt fungerar som baklängesiteratorer för vektorer. Exempel på användning:

```
int main() {
    int x[] = { 1, 3, 5, 7, 9 };
    for (rai<int> xit = rbegin(x, 5); xit != rend(x); ++xit) {
        cout << *xit << " ";
        *xit = 0;
    }
    cout << endl; // print 9 7 5 3 1
    pair<string, int> y[] = { make_pair("a",1), make_pair("b",2) };
    rai<pair<string, int> > yit;
    for (yit = rbegin(y, 2); yit != rend(y); ++yit) {
        cout << yit->first << "/" << yit->second << " ";
    }
    cout << endl; // print b/2 a/1
}
```

Implementera klassen `rai` och andra konstruktioner som behövs för att ovanstående program ska fungera. Den andra parametern till funktionen `rbegin` är antalet element i vektorn.

4. Implementera STL-funktionen `merge`, som samsorterar två sorterade följder. Funktionen har följande beskrivning (från STL-biblioteket i `libstdc++`):

```
/**
  *  @brief Merges two sorted ranges.
  *  @param  first1  An iterator.
  *  @param  first2  Another iterator.
  *  @param  last1   Another iterator.
  *  @param  last2   Another iterator.
  *  @param  result  An iterator pointing to the end of the merged range.
  *  @return  An iterator pointing to the first element "not less than" @a val.
  *
  *  Merges the ranges [first1,last1) and [first2,last2) into the sorted range
  *  [result, result + (last1-first1) + (last2-first2)).  Both input ranges
  *  must be sorted, and the output range must not overlap with either of
  *  the input ranges.  The sort is @e stable, that is, for equivalent
  *  elements in the two ranges, elements from the first range will always
  *  come before elements from the second.
 */
 template<typename InputIterator1, typename InputIterator2,
      typename OutputIterator>
   OutputIterator
   merge(InputIterator1 first1, InputIterator1 last1,
     InputIterator2 first2, InputIterator2 last2,
     OutputIterator result)
```

Observera: kommentarerna efter `@param result` och `@return` är uppgenbarligen felaktiga. De ska vara ”An output iterator” respektive ”An iterator pointing to the end of the merged range”.

5. I ett program läser man kommandon och heltal och summerar talen. Exempel på konversation med programmet (kommentarerna ingår inte; `p`, `a`, `u`, `c` och `r` är kommandon):

```
p       // print the sum, 0 to begin with
Sum is now 0
a 4     // add 4
a 3     // add 3
a 2     // add 2
p
Sum is now 9
u       // undo last add
u       // multiple undo's are allowed
p
Sum is now 4
c       // commit changes (make them permanent so they cannot be undone)
u       // nothing happens
a 3
a 2
p
Sum is now 9
r       // rollback, undo all changes since last commit
p
Sum is now 4
```

Programmet ser ut så här:

```
#include <iostream>
#include "accumulator.h"

using namespace std;

int main() {
    Accumulator accum;
    char cmd;
    while (cin >> cmd) {
        switch (cmd) {
            case 'p':
                cout << "Sum is now " << accum << endl;
                break;
            case 'a': {
                int nbr;
                cin >> nbr;
                accum += nbr;
                break;
            }
            case 'u':
                accum.undo();
                break;
            case 'c':
                accum.commit();
                break;
            case 'r':
                accum.rollback();
                break;
        }
    }
}
```

Implementera klassen `Accumulator`.