Examination
C++ Programming

2017–06–05, 08.00–13.00

Aid at the exam: one C++ book. The copies of the lecture slides are *not* allowed.

You must show that you know C++ and that you can use the C++ standard classes. "C solutions" don't give any points, even if they are correct.

Assessment (preliminary): the questions give $10 + 4 + 10 + 10 + 16 = 50$ points. You need 25 points for a passing grade (3/25, 4/33, 5/42).

---

1. Consider the following class:

```
class Ptr {
public:
    Ptr(int* p) : curr(p) {}
    int operator*() const { return *curr; }
private:
    int* curr;
};
```

The class just represents an `int` pointer and cannot be used for much, but the following will at least work:

```
int main() {
    int x[] = {1, 2, 3};
    Ptr p = x;
    std::cout << *p << std::endl;
}
```

a) Reading via the `Ptr` object works, but writing (`*p = ...;`) does not. Why? Modify the class such that this will work as well.

b) Modify the class such that the following code works:

```
int main() {
    int x[] = {1, 2, 3};
    for (Ptr<int> p = x; p != x + 3; ++p) {
        std::cout << *p << " ";
    }
    std::cout << std::endl;
    std::string y[] = {"Despite", "the", "constant", "negative", "press",
                       "covfefe"};
    for (Ptr<std::string> p = y; p != y + 6; ++p) {
        std::cout << *p << " ";
    }
    std::cout << std::endl;
}
```

2. The following Java program works fine:

```
class strplus {
   public static void main(String[] args) {
      int covfefe = 34567890;
      System.out.println("Covfefe count: " + covfefe);
   }
}
```

   The output will be (as expected):

```
Covfefe count: 34567890
```

   You can write a similar program in C++:

```
int main() {
   int covfefe_count = 34567890;
   std::cout << "Covfefe count: " + covfefe_count << std::endl;
 }
```

   The program is completely legal and compiles fine, but when I (Roger) tested it on my Mac I got the following output:

```
Segmentation fault: 11
```

   Explain what causes the output when the C++ program runs.

3. We want a class keeping track of names. We store the names in objects of the STL class set. We have chosen to use pointers in the set to represent the strings containing the names. The class looks like this:

```
// Reference to covfefe intentially omitted
class NameList {
public:
    NameList() {}
    ~NameList() {}
    void insert(const std::string& name) {
        names.insert(new std::string(name));
    }
    void printSorted() const {
        for (list_type::const_iterator it = names.begin(); it != names.end(); ++it) {
            std::cout << *it << std::endl;
        }
     }
private:
    typedef std::set<std::string*> list_type;
    list_type names;
};
```

   a) The class contains an obvious memory leak. Explain why the class leaks memory and change the class such that the eroor is corrected.

   b) The output in printSorted will not be at all as expected – it results in hexadecimal numbers instead of names. Why? Correct the function such that names are printed instead of numbers.

   c) The result is wrong even after the correction above – the names are not printed sorted despite the fact that a set is supposed to keep the elements sorted. Why not? Correct this error as well.

4. Write a function `is_strict_monotonic` that determines if a sequence of elements are strictly monotonic. By strict monotonic we mean that the elements either comes in a strictly increasing order or in a strictly decreasing order. Note that this also means that duplicate values cannot occur in a strict monotonic sequence. The function shall be able to work on all sequences that can be iterated over using an iterator and in which the elements can be pair-wise compared using the operator < (less than). The function shall return `true` if the sequence is strict monoton and `false` otherwise. You may assume that the sequence contains at least two elements.

   The function should work as in the following example (if you print a value of type `bool` one of 1 or 0 will be printed depending on if the value was `true` or `false`):

```
// Prints 1 (true)
int a[] = {1, 2, 3, 4};
std::cout << is_strict_monotonic(a,a+4) << std::endl;

// Prints 1 (true)
int b[] = {4, 3, 2, 1};
std::cout << is_strict_monotonic(b,b+4) << std::endl;

// Prints 0 (false)
int c[] = {1, 2, 7, 3, 4};
std::cout << is_strict_monotonic(c,c+5) << std::endl;

// Prints 0 (false)
std::vector<std::string> v = {"Despite", "the", "constant", "negative", "press",
                              "covfefe"};
std::cout << is_strict_monotonic(v.begin(),v.end()) << std::endl;

// Prints 1 (true)
sort(v.begin(),v.end());
std::cout << is_strict_monotonic(v.begin(),v.end()) << std::endl;
```

5. Write a program that reads a number of text files, whose names are given on the command line, and prints the 10 most common words together with the total number of occurancies of the words in all files. You may assume that the files only contains letters and whitespace. If a file cannot be opened, a suitable error message must be displayed.

   Exemple:

```
./count_words trump_tweets.txt dracula.txt
fake        9653
news        8965
the     8704
and     6272
to      5048
bad     4976
I       4792
of      4041
covfefe 3891
a       3456
```