CSE 333 ~~Final~~ 2nd Midterm Exam August 18, 2017

Sample Solution

Question 1. (20 points) STL. Complete the function `ChangeWords` below. This function has as inputs a vector of strings, and a map of <string, string> key-value pairs. The function should return a new `vector<string>` value (not a pointer) that is a copy of the original vector except that every string in the original vector that is found as a key in the map should be replaced by the corresponding value from that key-value pair.

Example: suppose that the vector `words` is `{"the", "secret", "number", "is", "xlii"}` and the map `subs` contains the pairs `{{"secret", "magic"}, {"xlii", "42"}}`. Then `ChangeWords(words, subs)` should return a new vector containing `{"the", "magic", "number", "is", "42"}`.

Hint: Remember that if `m` is a map, then referencing `m[k]` will insert a new key-value pair into the map if `k` is not already a key in the map. You need to be sure your code doesn't alter the map by adding any new key-value pairs. (Technical nit: `subs` is not a `const` parameter because you might want to use its `[]` subscripting operation in your solution, and `[]` is not a `const` function. It's fine to use `[]` as long as you don't actually change the contents of the map `subs`.)

Write your code below. Assume that all necessary headers and a `using namespace std;` directive have already been written for you.

```
vector<string> ChangeWords(const vector<string> &words,
                           map<string,string> &subs) {

  vector<string> result;
  for (const auto &word: words) {
    if (subs.count(word) != 0) {
      result.push_back(subs[word]);
    } else {
      result.push_back(word);
    }
  }
  return result;
}
```

There are, of course, many other possible solutions and, if done correctly, those received full credit.

Sample Solution

Question 2. (12 points) More STL – the evil, tricky question on this exam. Here is a little program that has a small class `Thing` and a main program (necessary `#includes` and `using namespace std;` omitted to save space).

```
class Thing {
public:
  Thing(int n): n_(n) { }
  int getThing() const
                { return n_; }
  void setThing(int n){n_=n;}
private:
  int n_;
};
```

```
int main() {
  Thing t(17);
  vector<Thing> v;
  v.push_back(t);
}
```

This code compiled and worked as expected. But then we added the following two lines of code (plus the appropriate `#include <set>`):

```
set<Thing> s;
s.insert(t);
```

The second line (`s.insert(t)`) failed to compile and produced dozens of spectacular compiler error messages, all of which looked more-or-less like this (edited to save space):

```
In file included from string:48:0, from bits/locale_classes.h:40,
from bits/ios_base.h:41,from ios:42,from ostream:38, from
/iostream:39,from thing.cc:3: bits/stl_function.h: In
instantiation of 'bool std::less<_Tp>::operator()(const _Tp&,
const _Tp&) const [with _Tp = Thing]': <<many similar lines
omitted>> thing.cc:37:13: required from here bits/stl_function.h:
387:20: error: no match for 'operator<' (operand types are 'const
Thing' and 'const Thing') { return __x < __y; }
```

What on earth is wrong? Somehow class `Thing` doesn't work with `set<Thing>` even though `insert` is the correct function to use here. (a) What is the most likely reason, and (b) what would be needed to fix the problem? (Be brief but precise – you don't need to write code in your answer, but you can if that helps make your explanation clear.)

(a) Items can only be stored in a set if it is possible to compare them to each other. STL containers compare elements using `operator<`, and that function hasn't been provided for `Thing`.

(b) The fix is to add an appropriate `operator<` as either a member function in `Thing`, or as a free-standing function that compares two `Thing&` parameters.

Question 3. (24 points) The usual, demented, dreaded virtual function madness. Consider the following program, which does compile and execute with no errors, except that it leaks memory (which doesn't matter for this question).

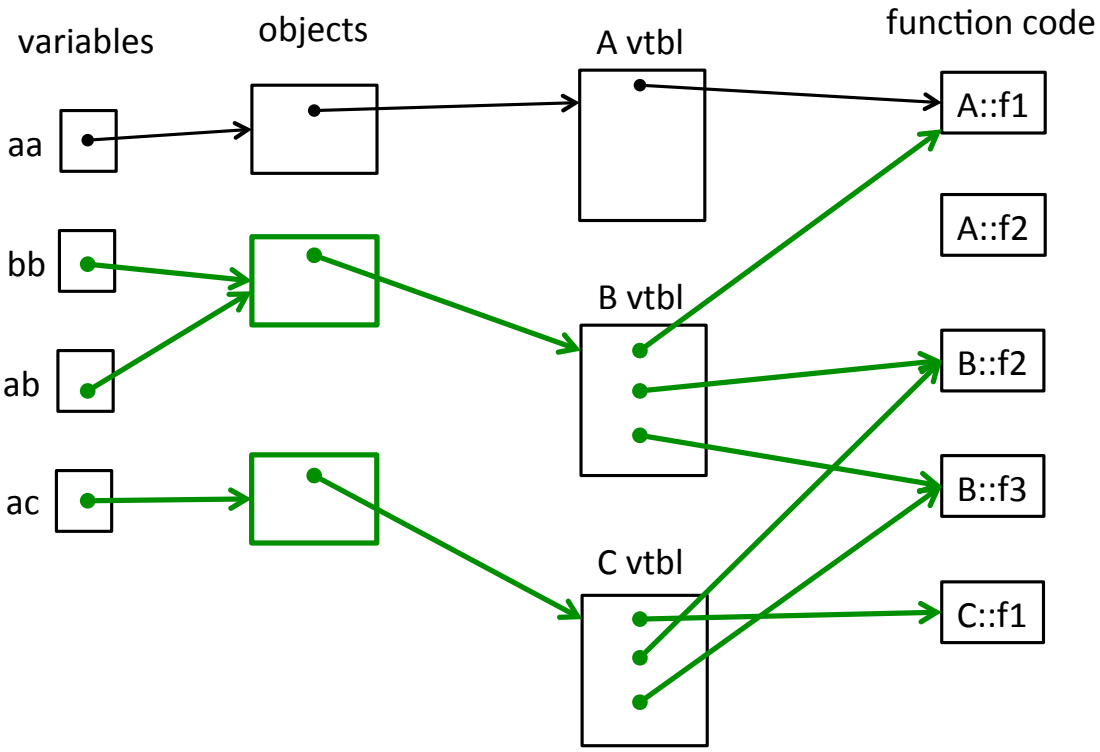
```
#include <iostream>
using namespace std;

class A {
public:
  virtual void f1() { f2(); cout << "A::f1" << endl; }
  void f2() { cout << "A::f2" << endl; }
};
class B: public A {
public:
  virtual void f3() { f1(); cout << "B::f3" << endl; }
  virtual void f2() { cout << "B::f2" << endl; }
};
class C: public B {
public:
  void f1() { f2(); cout << "C::f1" << endl; }
};
int main() {
  A* aa = new A();
  B* bb = new B();
  A* ab = bb;
  A* ac = new C();
  aa->f1();
  cout << "----" << endl;
  bb->f1();
  cout << "----" << endl;
  bb->f2();
  cout << "----" << endl;
  ab->f2();
  cout << "----" << endl;
  bb->f3();
  cout << "----" << endl;
  ac->f1();
  return 0;
}
```

Remove this page from the exam, then answer questions about the program on the next page.

Sample Solution

Question 3. (cont.) (a) (12 points) Complete the diagram below by adding the remaining objects and all of the additional pointers needed to link variables, objects, virtual function tables, and function bodies. Be sure that the order of pointers in the virtual function tables is clear (i.e., which one is first, then next, etc.). One of the objects and a couple of the pointers are already included to help you get started.



Notes: The vtable for `B` could reverse the order of the `f2` and `f3` pointers – that is arbitrary. But the same order must then be used in `C`'s vtable. The `f1` pointer must appear first in all three vtables. No vtable contains a pointer to `A::f2` since that function is not virtual.

(b) (12 points) Write the output produced when this program is executed. If the output doesn't fit in one column in the space provided, write multiple vertical columns showing the output going from top to bottom, then successive columns to the right.

```
A::f2
A::f1
----
A::f2
A::f1
----
B::f2
----
```

```
A::f2
----
A::f2
A::f1
B::f3
----
B::f2
C::f1
```

Sample Solution

Question 4. (18 points) A bit of concurrency. Here is yet another small program that creates a handful of threads and lets them run concurrently. (Header files and `usings` omitted to save space, but this does compile and execute without crashing.)

```
const int NUM_THREADS = 3;
static int result = 0;          // global var. changed by threads
static pthread_mutex_t lock;  // unused variable (for now)
struct t_args {  // arguments for one thread:
  int amount;     //   amount to add to global result
  int count;      //   number of times to add amount to result
};
// Given a t_args struct, add amount to result,
// repeated count times.
void *update_result(void *arg) {
  struct t_args *p_arg = (struct t_args *) arg;
  for (int i = 0; i < p_arg->count; i++) {
    pthread_mutex_lock(&lock);      // added for part (c)
    result += p_arg->amount;
    pthread_mutex_unlock(&lock);    // added for part (c)
  }
  return NULL;
}
int main(int argc, char** argv) {
  struct t_args args[NUM_THREADS] = {{5, 4}, {20, 6}, {-5, 2}};
  pthread_t thrds[NUM_THREADS];
  // initialize lock
  pthread_mutex_init(&lock, NULL);   // added for part (c)
  // create threads
  for (int i = 0; i < NUM_THREADS; i++) {
    if (pthread_create(&thrds[i], NULL, &update_result, &args[i])
                                                          != 0) {
      cerr << "pthread_create failed" << endl;
    }
  }
  // wait for threads to finish
  for (int i = 0; i < NUM_THREADS; i++) {
    if (pthread_join(thrds[i], NULL) != 0) {
      cerr << "pthread_join failed" << endl;
    }
  }
  // destroy mutex (good practice but not required for static
  // mutex variables - only required for dynamic or automatic)
  pthread_mutex_destroy(&lock);     // added for part (c)
  // print final value of result and exit
  cout << "Total: " << result << endl;
  return 0;
}
```

(continued on the next page, but leave this page in the exam. You may need to write some changes in the above code.)

Question 4. (cont) When we run this program it starts three threads, waits for them all to finish, and then prints the final value of the variable `result`.

(a) (2 points) What value should this program print if the threads do not interfere with each other (i.e., if, for example, the three threads were executed sequentially, one after the other, rather than running concurrently):

Expected value if no interference: **130**

(b) (6 points) When the threads run concurrently, what is the possible range of values that the program could print? (i.e., what are the possible final values for variable `result`?) Give the minimum and maximum possible values. If concurrent execution always produces the same output, just list that value as both the minimum and maximum.

(Hint: to answer this part of the question, it helps to assume that if we execute the program repeatedly, the concurrent thread scheduling will occur in as many different, unpredictable ways as possible.)

Minimum possible value: **-10**

Maximum possible value: **140**

Reason (not required): Since there is no synchronization, any of the updates can be lost if one thread reads the value of `result` and then another thread writes a new value before the first thread can write its updated value. The second write will replace the first one, and the update from the other thread will be lost.

(c) (10 points) If it is possible for concurrent execution to produce different results, show the minimal changes needed to guarantee that the global variable `result` has the expected final value that it would have if the threads executed sequentially (i.e., the value given in your answer to part (a)). Your solution should, however, still allow for as much concurrent execution of the different threads as possible. For example, it is not reasonable to rewrite the code so the threads run sequentially, one after the other or otherwise rewrite the existing code so that it is substantially different.

Show your changes by writing them on the code on the previous page.

A lock is needed to to synchronize updates to `result`. Code changes shown in bold green on the previous page.

A couple of shorter-answer questions to finish up.

Question 5. (12 points) Reference counting. The STL `shared_ptr` smart pointers use a strategy known as *reference counting* to decide when to delete the object whose pointer is owned by the `shared_ptr`.

(a) (6 points) Very briefly, what is *reference counting*? (You can draw diagrams if you wish to help illustrate your answer, but that is not required.)

For each object, we store a count of the number of pointers currently referring to that object. When pointers are changed we update the reference count(s) of the associated object(s). Whenever a reference count is decreased to 0 we know that no pointers refer to the object, so it is deleted.

(b) (6 points) Reference counting is one strategy for deleting resources when they are no longer needed. Is it suitable as a general memory management strategy (i.e., could we use it instead of algorithms like garbage collection, which is used in Java implementations)? Why or why not? (Please be brief and to the point)

Reference counting cannot be used as a general memory management strategy because it cannot properly handle data structures that contain cycles. If a cycle is present, all objects in the cycle will have non-zero reference counts because of the references to each other. Those reference counts will never decrease to zero, even if no other variable points to any of the objects, so those objects will never be deleted.

(It also is true that reference counting has a fairly substantial overhead for simple pointer operations because we need to adjust reference counts whenever we modify a pointer. But if that were the only problem, we could still use reference counting if we were wiling to tolerate the overhead. The inability to handle cycles means that reference counting cannot be used as the sole memory management strategy.)

Question 6. (12 points, 2 each) Recall that the network protocol stack is organized in a sequence of layers. Listed alphabetically they are:

- Application (HTTP, etc.)
- Data link
- Network (IP)
- Physical
- Transport (TCP, UDP, etc.)

For each of the following, identify the layer in the network stack that performs the described operation. Be sure you identify the layer, not the protocol (e.g., IMAP is a protocol, but that does not necessarily identify a particular layer). If it matters for a particular question, you can assume we are using TCP sockets for stream connections.

Note: Answers only needed to include the layer name. The protocol names are shown here for additional information, but were not required.

(a) Transmit an Ethernet packet on the local network from one host machine's NIC interface address to another's.

Data link

(b) Route an incoming message to the correct port to handle a particular service (e.g., web pages, ssh remote login, email, etc.)

Transport (TCP)

(c) Forward a packet from the local wired or wireless network to a different local network if its destination address is not on the same local network.

Network (IP)

(d) If two packets that make up a message arrive out of order, rearrange them into the correct order before they are transmitted to the process reading the data.

Transport (TCP)

(e) Send a `200 OK` message to a web browser in reply to a request for something from a web server.

Application (HTTP)

(f) Discover whether `www.erehwon.com` is a known Internet domain name.

Application (DNS)

Sample Solution

Question 7. (2 free points – all answers get the free points)

Draw a picture of something that puts you in a good mood, maybe something that you will enjoy once the summer quarter is over and you have some time to relax. (But keep it suitable for a family audience – you don't know who might read this! ☺)

Congratulations on lots of great work this summer !!
Have a great vacation and say hello when you get back !
The CSE 333 staff