CSE 333 22sp Final Exam 6/8/22 Sample Solution

Question 1. (15 points) C++ STL. In these days of product shortages, it's hard to keep track of what is currently available where. We have a C++ `map` that contains information about stores and products currently available at those stores. The keys in the map are strings giving store names. The associated values are lists of items available in those stores. For example:

```
map<string, vector<string>> inventory =
  { {"qfc", {"bread", "milk", "bananas", "soap"}},
    {"bartells", {"soap", "toothpaste"}},
    {"safeway", {"milk", "bread", "bananas", "hotdogs"}}
  };
```

We would like to write a function that uses this information to produce a map whose keys are the items (values) from the above data and whose values are a set of stores where those items are available. The first few entries in the result produced from the above data might be the following:

```
{ {"bread",   {"qfc", "safeway"}},
  {"soap",    {"bartells", "qfc"}},
  {"hotdogs", {"safeway"}},
  ... }
```

The keys in the maps and the store names in the result sets can be in any order – your code does not need to guarantee any particular ordering of these.

Complete the definition of function `find_stores` on the next page so it produces a map of products and stores where they are available from an input map of stores and products that are available in those stores. Although the code is fairly short, please write it on the next page to ensure there is plenty of space for your answer, rather than trying to fit it at the bottom of this page.

(Note: since the store names are keys in the original map, we are guaranteed that they are unique, so we do not have to worry about duplicate values in the result sets.)

(Reminder: reference information about STL containers is included in the tear-off pages at the end of the exam.)

Write your

Answer

On the

Next

Page…

Question 1. (cont.) Complete the definition of function `find_stores` below. You should assume all necessary headers have already been `#included`, and that a `using namespace std;` directive has been supplied so you do not need to write `std::` in front of all standard library names. You almost certainly will not need all of this space.

```
// return a map from product names to sets of stores where those
// products are available
map<string, set<string>> find_stores(
                   map<string, vector<string>> inventory) {
// write your implementation below

  map<string, set<string>> product_locations;

  // iterate through each item in the inventory
  for (auto& store_list : inventory) {
    string store = store_list.first;
    vector<string> items = store_list.second;

    // add the stores for each item to product_locations
    for (auto& item : items) {
      // Note: product_locations[item] automatically instantiates
      // a set for new items
      product_locations[item].insert(store);
    }
  }
  return product_locations;

}
```

CSE 333 22sp Final Exam 6/8/22 **Sample Solution**

Question 2. (18 points) Templates and things. The following header file defines a class that holds a pair of integers and includes a constructor and functions for accessing the values.

```
#ifndef PAIR_H_
#define PAIR_H _

template <typename T>    // <class T> also works
class Pair {
 public:
  // Construct a Pair with given first and second values
  Pair(~~int~~ T first, ~~int~~ T second)
      : first_(first), second_(second) { }

  // accessors: return first and second items from Pair
  ~~int~~ T first()  const { return first_;  }
  ~~int~~ T second() const { return second_; }

 private:
  // instance variables
  ~~int~~ T first_;
  ~~int~~ T second_;
};

#endif  // PAIR_H _
```

(a) (5 points) We would like to generalize this class so it can be used to store any pairs of values as long as both values have the same type (i.e., pairs of `int`s, pairs of strings, etc.)

Show the changes needed to make this a generic class where the element type is a type parameter instead of `int`. You should write your changes and additions in the above code.

(Changes shown above in bold green.)

(continued on the next page)

Question 2. (cont.) We would now like to add an addition (+) operator to the generic `Pair` class on the previous page. If (a,b) and (c,d) are `Pair` values, then (a,b)+(c,d) should yield a new `Pair` containing (a+c, b+d). Neither of the original `Pair` objects should be modified. You do not need to check whether addition (+) is defined on the items stored in a `Pair` – that is handled for you by the compiler when the addition operator is used. The compiler will check that the actual type used when the template is instantiated supports addition and produce appropriate error messages if it does not.

(b) (5 points) Write the function declaration (not the implementation) to be added to the header file `Pair.h` for the new `operator+`. If it is possible to add `operator+` as either a member or non-member function of the `Pair` class, you can use whichever one you prefer.

Member template function version in `<typename T> class Pair`:

```
   Pair<T> operator+(const Pair<T> &other) const;
```

Non-member template function version:

```
   template <typename T>
   Pair<T> operator+ (const Pair<T> &a, const Pair<T> &b);
```

(either version was acceptable)

(c) (8 points) Give the code to implement this new addition operator as it would appear in a separate file `Pair.cc` containing definitions of functions not implemented in the header `Pair.h`.

(Note: if the code is part of a template, it probably would not be in a separate `.cc` file, but would be included in the `.h` file containing the template declaration. The main difference is that if the operator is defined as a member function in the class template, it should not have a separate declaration of the type parameter `T`, and if the body of the function is included as part of the declaration, the `Pair<T>::` scope operation would not be needed either. If it is a non-member function it would appear as shown here, even when it is included in the header file.)

Member function version in separate `.cc` file:

```
template <class T>
Pair<T> Pair<T>::operator+(const Pair<T> &other) const {
  return Pair<T>(first() + other.first(),
                 second() + other.second());
}
```

Non-member function version:

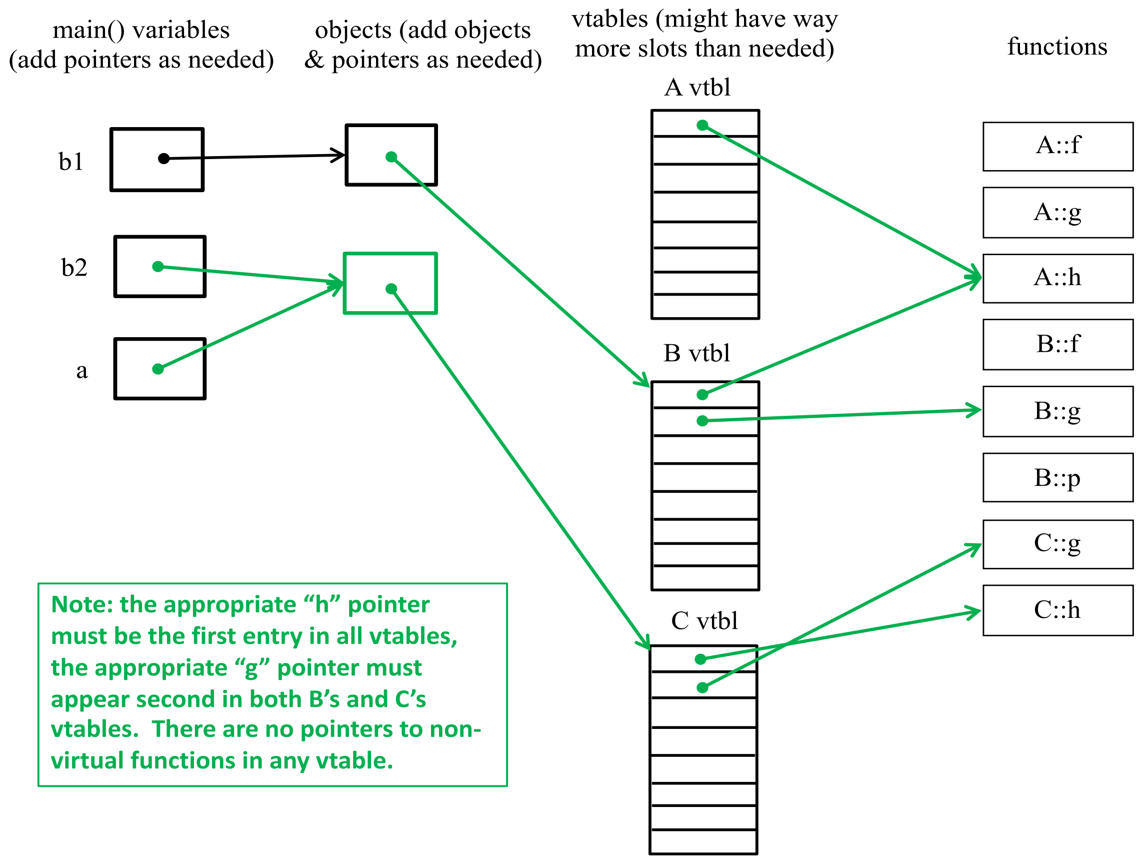
```
template <typename T>
Pair<T> operator+ (const Pair<T> &a, const Pair<T> &b) {
  return Pair<T>(a.first() + b.first(),
                 a.second() + b.second());
}
```

Question 3. (20 points) Here we go again – dynamic dispatch and friends. As usual this program compiles and executes with no errors. Headers and `using namespace std` omitted to save space.

```
class A {
 public:
          void f() { h(); cout << "A::f" << endl; }
          void g() {      cout << "A::g" << endl; }
  virtual void h() {      cout << "A::h" << endl; }
};

class B: public A {
 public:
          void f() {      cout << "B::f" << endl; }
  virtual void g() { h(); cout << "B::g" << endl; }
          void p() { g(); cout << "B::p" << endl; }
};

class C: public B {
 public:
          void g() { f(); cout << "C::g" << endl; }
  virtual void h() { p(); cout << "C::h" << endl; }
};

int main() {
  cout << "--part 1--" << endl;
  B* b1 = new B();
  b1->f();
  cout << "---" << endl;
  b1->g();
  cout << "---" << endl;
  b1->h();
  cout << "---" << endl;
  b1->p();
  cout << "--part 2--" << endl;
  B* b2 = new C();
  A* a = b2;
  a->f();
  cout << "---" << endl;
  a->g();
  cout << "--part 3--" << endl;
  b2->f();
  cout << "---" << endl;
  b2->g();
  return 0;
}
```

Continue with the problem on the next pages. **Do not remove this page from the exam.**

Question 3. (cont.) (a) (6 points) Complete the diagram below to show all of the variables, objects, virtual method tables (vtables) and functions in this program. Parts of the diagram are supplied for you.



Note: the appropriate "h" pointer must be the first entry in all vtables, the appropriate "g" pointer must appear second in both B's and C's vtables. There are no pointers to non-virtual functions in any vtable.

(b) (14 points) What does this program print when it executes? (write your answer in multiple columns if needed)

```
--part 1--
B::f
---
A::h
B::g
---
A::h
---
A::h
B::g
B::p
```

```
--part 2--
B::f
C::g
B::p
C::h
A::f
---
A::g
--part 3--
B::f
---
B::f
C::g
```

Question 4. (18 points) More C++ classes. Consider the following program, which contains a simple integer wrapper class `Int` and a main function that uses it. Header `#includes` and `using namespace std;` omitted to save space. In the box on the right, write the output that is produced when this program is run. It compiles and executes with no errors.

```
class Int {
 public:
  // Constructors & destructor
  Int()                  { cout<<"Int()"  <<endl; val_=17; }
  Int(int n)             { cout<<"Int(n)" <<endl; val_=n;  }
  Int(const Int &other){cout<<"copy ctr"<<endl; val_=other.val_;}
  ~Int()                 { cout<<"dtr"<<endl; }

  // accessor function
  int get() { cout<<"Int.get"<<endl; return val_; }

  // assignment
  Int &operator=(const Int& rhs) {
    cout<<"Int.op="<<endl;
    if (this == &rhs) return *this;
    this->val_ = rhs.val_;
    return *this;
  }
 private:
  int val_;   // instance variable
};  // end of class Int

int IntSum(Int v[], int sz) {
  int result = 0;
  for (int k = 0; k < sz; k++) {
    result += v[k].get();
  }
  return result;
}

int main() {
  Int x = 10;
  Int y = x;
  Int a[2];
  cout << "---" << endl;
  a[0] = 42;
  cout << "---" << endl;
  int sum = IntSum(a, 2);
  cout << "sum = " << sum << endl;
  cout << "---" << endl;
  Int* p;
  p = &x;
  int n = p->get();
  cout << "p->get() = " << n << endl;
  return EXIT_SUCCESS;
}
```

Program output:

```
Int(n)
copy ctr
Int()
Int()
---
Int(n)
Int.op=
dtr
---
Int.get
Int.get
sum = 59
---
Int.get
p->get() = 10
dtr
dtr
dtr
dtr
```

Question 5. (14 points) Networking. Below is the pseudo-code for a very simple TCP server that accepts connections from clients and exchanges data with them. However, this code doesn't work because it has structural errors. In particular, some functions are called at the wrong time or in the wrong place, and there may be other problems. Write in corrections below to show how the pseudo-code should be rearranged, changed, or fixed to have the proper structure for a simple server. Feel free to draw arrows showing how to move code around, but be sure it is clear to the grader what you mean.

You should assume that all functions always succeed – ignore error handling for this question. Further, assume that the first address returned by `getaddrinfo` works and we don't need to search that linked list to find one that does work. Also, ignore the details of parameter lists – assume that all the "…" parameters are valid and appropriate.

Reminder: there is some potentially useful reference information at the end of the exam.

```
int main(int argc, char **argv) {
  struct addrinfo hints, *rp;
  memset(&hints, 0, sizeof(hints));
  hints.ai_... = ...;        // specify values for options
  getaddrinfo(NULL, argv[1], &hints, &rp);

  // ok to assume *rp is a valid address and will work here
  int fd = socket(rp->ai_family,
                  rp->ai_socktype, rp->ai_protocol);
  setsockopt(fd, SOL_SOCKET, SO_REUSEADDR, ...);
  freeaddrinfo(rp);
  while (1) {

    bind(fd, rp->ai_addr, rp->ai_addrlen);

    fd = accept(fd, ...);

    listen(fd, SOMAXCONN);

    // talk to client as needed
      read(fd, ...);
      write(fd, ...);
    close(fd);
  }
  return EXIT_SUCCESS;
}
```

(See solution on next page)

Question 5 solution. Here is a corrected version of the pseudo-code, written out in full for clarity. Test answers were only expected to show these changes written on the same page as the original code, with appropriate indications of where to insert, delete, or move things. The major changes needed are:

- **Use separate int file descriptors for the listening and client sockets.**
- **Move the code to bind and open the listening socket outside the loop so it is done once as part of the server initialization.**
- **Only close the client file descriptor inside the loop; leave the listening fd open so it can be used to accept the next client.**

The original code did not have any way to shut down the server by exiting the main loop, so there would be no reason to close the listening file descriptor. However, if a `close(listen_fd)` were included in the corrected code, it would have to be at the end outside the loop.

Note that there are some variations on this code that are also correct. For instance, the `freeaddrinfo(rp);` function call can appear after the `listen` operation.

```
// pseudo-code for a network server
int main(int argc, char **argv) {
  struct addrinfo hints, *rp;
  memset(&hints, 0, sizeof(hints));
  hints.ai_... = ...;   // specify values for desired options
  getaddrinfo(NULL, argv[1], &hints, &rp); // assume *rp works
  int listen_fd = socket(rp->ai_family,
                         rp->ai_socktype, rp->ai_protocol);
  setsockopt(listen_fd, SOL_SOCKET, SO_REUSEADDR, ...);
  bind(listen_fd, rp->ai_addr, rp->ai_addrlen);
  freeaddrinfo(rp);
  listen(listen_fd, SOMAXCONN);
  while (1) {
    int client_fd = accept(listen_fd, ...);
    // talk to client as needed
      read(client_fd, ...);
      write(client_fd, ...);
    close(client_fd);
  }
  close(listen_fd);
  return EXIT_SUCCESS;
}
```

Question 6. (14 points, 2 each) Too many things at once… Consider the following C program using pthreads, which does compile and execute without errors (headers omitted to save space):

```
int x = 0;

void * thread_worker(void * ignore) {
  x = x + 1;
  printf("x=%d\n", x);
  return NULL;
}

int main() {
  pthread_t t1, t2;
  int ignore;
  ignore = pthread_create(&t1, NULL, &thread_worker, NULL);
  ignore = pthread_create(&t2, NULL, &thread_worker, NULL);
  pthread_join(t1, NULL);
  pthread_join(t2, NULL);
  printf("final x=%d\n", x);
  return EXIT_SUCCESS;
}
```

For each of the following output sequences, circle "yes" if it could be produced by some possible execution of the above program and circle "no" if it could never happen under any circumstances. (Hint: at least one of these sequences is possible.)

YES [✓ NO] `x=0 x=1 final x=1`

[✓ YES] NO `x=1 x=1 final x=1`

YES [✓ NO] `x=1 x=2 final x=1`

YES [✓ NO] `x=1 x=1 final x=2`

[✓ YES] NO `x=1 x=2 final x=2`

[✓ YES] NO `x=2 x=1 final x=2`

[✓ YES] NO `x=2 x=2 final x=2`

(Note: in these answers we assumed that each `printf` operation successfully prints its complete output without interruption from another thread.)

Question 7. (1 free point – all answers get the free point) Draw a picture of something you plan to do this summer!

Congratulations on lots of great work this quarter!!
Have a great summer!
The CSE 333 staff