Question 1. (18 points) The traditional C++/STL programming. We have accumulated a list of stores and products that they sell. We'd like to write a small C++ program that reads this information from the standard input file (cin) and then prints out a list of all the products sorted by product name and for each shows the stores that sell them.

Example: suppose we have the following input read from cin:

```
qfc banana
amazon book
amazon iphone
cvs toothpaste
qfc toothpaste
apple iphone
apple computer
ubookstore book
amazon banana
```

After reading this input and reaching the end of file on cin, the program should produce the following output:

```
banana: qfc amazon
book: amazon ubookstore
computer: apple
iphone: apple amazon
toothpaste: cvs qfc
```

In the output, the products must be listed (sorted) in order, but the stores that sell each product can be listed in any order.

Since this is an exam question, you may assume that input operations like reading strings will succeed, and you should use cin>>s to read the strings and skip whitespace. You can assume the input has exactly the right format and does not have extra strings or missing data, and the store and product names consist only of alphabetic characters. You may also assume there are no duplicate "store product" entries in the input data. Strings must match exactly to be equal – you do not need to convert strings to lower-case letters or anything like that – i.e., keep it simple.

Write your code in the space on the following page. You may assume that all necessary #include statements are provided for whatever standard libraries you wish to use, but you do need to declare appropriate variables to hold data as you read and process it.

Hints: various STL containers like map might be particularly useful. Basic reference information about STL is included on the last page of this exam.

(continued on next page)

Question 1. (cont.) Write your code for this problem below:

```
// assume all necessary libraries are #included
using namespace std;
int main(int argc, char** argv) {
 // add your code below
 map<string, vector<string>> product offerings;
  string store, product;
  // build up the map
 while (cin >> store && cin >> product) {
   product offerings[product].push back(store);
  }
  // print results
  for (const auto& element : product offerings) {
    string prod = element.first;
   vector<string> stores = element.second;
    cout << prod << ":";</pre>
    for (const auto& s : stores) {
      cout << " " << s;
    }
   cout << endl;</pre>
  Ł
 return EXIT SUCCESS;
}
```

Note: There are, of course, many ways to solve the problem. Correct solutions received full credit even if they did not exactly match this code.

Question 2. (18 points) A different twist on the usual inheritance question. Consider the following program and answer questions about it on the next page. This program prints some of the things that someone playing the card game poker might say during the game. But don't worry – you don't need to know anything about poker to answer this question.

```
#include <iostream>
#include <cstdlib>
using namespace std;
class Poker {
 public:
 /* to be supplied if needed */
 virtual void m1() {std::cout << "You're ";}</pre>
          void m2() {std::cout < "all ";}</pre>
};
class Hands : public Poker {
public:
          void m1() {std::cout << "I'm ";}</pre>
 virtual void m3() {std::cout << "not ";}</pre>
};
class Flush : public Hands {
public:
 virtual void m2() {std::cout << "so ";}</pre>
          void m3() {std::cout << "in!";}</pre>
};
int main(int argc, char** argv) {
 Poker* ph = new Hands();
 Poker* pf = new Flush();
 Hands* hf = new Flush();
 ph->m1();
 pf->m2();
 hf->m3();
  return EXIT SUCCESS;
}
```

Continue with the problem on the next page.

Question 2. (cont.) (a) (4 points) When we try to run the program on the previous page, it won't compile. What, precisely, are the errors in the code that prevent it from compiling?

The two function calls ph->m1() and ph->m2() in main won't compile because class Poker does not have functions named m1 or m2.

(b) (6 points) On the previous page, make changes to the code so that it will compile and so it will print out the following three strings when it is executed: I'm all in!. You may not change or delete any of the functions currently in the program – in particular, they must continue to print whatever they print in the original code. You need to decide what new function(s) to add, if any, what they should print, and whether they are virtual or not. Hint: you may find it helpful to work on this part of the question while also working on part (c) below.

See new functions added to class Poker on the previous page in bold green type. Function m1 must be virtual and can do anything. Function m2 must not be virtual and must print "all".

(c) (8 points) Complete the diagram below to show all of the variables, objects, virtual method tables (vtables) and functions in the program after making your changes in part (b). Parts of the diagram are supplied for you.



Question 3. (10 points) We know that C++ smart pointers can handle some memory management tasks for us automatically. However, they have to be used properly.

(a) (5 points) Suppose we have the following Node definition, which uses shared_ptrs for links used to create double-linked lists and similar data structures.

```
struct Node {
    int data;
    shared_ptr<Node> next;
    shared_ptr<Node> prev;
};
```

Will this prevent memory leaks if we have a shared_ptr variable pointing to the first node in a list? Give a brief justification or example for your answer.

No. Once there is more than one node in a double-linked list, each of the nodes will have a reference count of at least 1 because of references from either its predecessor or successor node or both. Even if we delete all other smart pointers referencing the list nodes, the reference counts will never be decremented to 0, so the nodes will never be deleted.

(b) (5 points) One of our colleagues says we should use weak_ptrs instead of shared_ptrs for the backward (prev) links in the nodes:

```
struct Node {
    int data;
    shared_ptr<Node> next;
    weak_ptr<Node> prev;
};
```

Will this prevent memory leaks if we have a shared_ptr variable pointing to the first node in a list? Give a brief justification or example for your answer.

No. It will solve the reference count problem caused by the combination of forward and backward links in a double-linked list. However, it is still possible to create cycles in, for example, a circular double-linked list, and that will prevent reference counts from reaching 0 for nodes that are in the cycle.

Question 4. (12 points) Templates and Boxes. Here is a small program that defines and uses a class Box that holds items that have integer weights. Each Box has a maximum weight capacity that is set when it is created, and we keep track of the remaining capacity as items are added to the Box.

```
#include <iostream>
#include <cstdlib>
using namespace std;
/* A box with a fixed weight limit */
Template <typename T> class Box {
 public:
  // Construct new Box with default max weight 10
  Box() : remaining weight (10) { }
  // Construct new Box with given max weight
  Box(int T w): remaining weight (w) { }
  // Add item with given weight w to this Box.
  // Return 1 (success) if successful, return 0 if not enough
  // capacity in the box for the new item.
  int add(int T w) {
    if (w <= remaining weight ) {
                                         Minimal changes needed are shown in bold green.
      remaining weight -= w;
                                         In main, when declaring b, we need to specify
      return 1;
                                         Box<double> or change the initial constant to 5.0,
    } else {
                                         otherwise the b (5) initialization will create a
      return 0;
                                         template using int for the type parameter T.
    }
  }
  // return remaining weight capacity in this box
  int T remaining() {
    return remaining weight ;
  }
private:
  int T remaining weight ; // remaining weight capacity in this Box
};
/* main program to use a Box */
int main(int argc, char** argv) {
 Box<double> b(5);
  cout << b.add(3) << endl;</pre>
  cout << b.remaining() << endl;</pre>
  return EXIT SUCCESS;
}
```

We've decided that this Box class would be much more useful if would work with weight values that have any arithmetic type that supports the necessary operations (<=, -=, etc.). Write in changes to the above code to make this into a class template where the weight values could be any suitable type T, and adjust the main program to create a Box with weights that have values of type double by using appropriate type parameter values if needed.

Question 5. (18 points) Sockets & strings – Oh My! We're working on a web program and need to implement a function that reads bytes from a socket and locates and returns a data string extracted from the input stream. We assume the data that we want is immediately preceded by the contiguous (no spaces) characters #start# and immediately followed by the characters #end#. So, for example, if the input stream is

Stuff #start# #stop# this! is not the #end # yet#end# stuff

then the function should return all of the characters between the markers, including any whitespace or special characters, but not including the markers. The data returned from this stream is underlined above, and includes an initial space that immediately follows #start#. Note that the data string can contain some of the characters that can appear in the marker #end#, but only the first complete occurrence of #end# marks the data end.

Assumptions: You may assume that the input stream will contain exactly one copy of the markers #start# and #end# with zero or more characters between them. The characters between the markers in the stream should be returned as a C++ string, which might be an empty string if there are no characters between #start# and #end#.

As with the hw4 server, you will need to use the POSIX read function to read bytes from the stream, and your code needs to handle possible error return values from read, including EAGAIN and EINTR. Also, as with hw4, it is possible that a single read operation may return only part of the data. In particular, a single read might end with only part of the #start# or #end# markers, with the rest of the #start# or #end# marker appearing in the input the next time we call read.

Once your function has read all of the characters surrounded by the #start# and #end# markers it should return a string containing those characters, without the markers, as its result. You do not need to close the stream or do anything else with it. However, if it simplifies things, you can read as much additional data from the stream as you wish – maybe even reading until reaching end of file and then extracting the message – whatever is easiest.

You should assume that all necessary library function headers have been #included, and that using namespace std; has also been included in the code. You do not need to write these.

Hint: the information about C++ strings on the reference page at the end of the exam might be useful.

(continued on next page)

Question 5 (cont.) Write your implementation of function ReadData below.

```
/* Read from the socket file descriptor sfd and return
   a string containing the characters that are surrounded
  by the markers #start# and #end#. The markers are not
   included in the returned string. */
string ReadData(int sfd) {
  constexpr int kBufSize = 10; // small length for testing
  char buf[kBufSize];
  string data;
  int chars read = 0;
  // read from the socket until end of file
  while ((chars read = read(sfd, &buf, kBufSize-1)) != 0) {
    if (chars read == -1) {
      if (errno != EINTR && errno != EAGAIN) {
        perror("failed to read from socket");
        exit(EXIT FAILURE);
      ł
      continue;
    }
   buf[chars read] = \sqrt{0'};
    data += string(buf);
  }
  // calculate indices for the data and return data
  size t start = data.find("#start#") +
                    sizeof("#start#");
  size t end = data.find("#end#", start);
  size t data len = end - start;
  return data.substr(start, data len);
}
```

Note: as with other programming problems, there are multiple correct ways of solving this problem. In particular, it's fine to stop reading from the stream after reading the complete #end# marker, although this solution, where we just read everything until the end of the input, might be a bit simpler.

If a non-recoverable error is returned by read, the only reasonable thing to do is to terminate the program. The function specification doesn't allow us to return an empty string in that case, because there is no way to distinguish that from a legitimate empty message between #start# and #end#.

Question 6. (16 points, 2 points each part) Concurrency. Consider the following C program that creates a pair of threads and executes them. This program does compile and execute without errors.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
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.)

(a) YES NO
$$x=0$$
 $x=1$ final $x=1$
(b) YES NO $x=1$ $x=1$ final $x=1$
(c) YES NO $x=1$ $x=2$ final $x=1$
(d) YES NO $x=1$ $x=1$ final $x=2$
(e) YES NO $x=1$ $x=2$ final $x=2$
(f) YES NO $x=2$ $x=2$ final $x=1$
(g) YES NO $x=2$ $x=1$ final $x=2$
(h) YES NO $x=2$ $x=2$ final $x=2$

Question 7. (16 points) The question we dropped from the midterm to save time! One of your colleagues has some questions about C++ classes. Their example is the following code, which doesn't do very much, but does include the required constructors, destructor, and assignment for a typical class:

```
#include <cstdlib>
class Ogre {
 public:
  Ogre() { name = new char[10]; }
  Ogre(const Ogre &other) = default;
  Ogre& operator=(const Ogre &other) = default;
  ~Ogre() { delete [] name ; }
 private:
  char* name ;
};
int main(int argc, char **argv) {
                                     // Line 1
  Ogre shrek;
                                     // Line 2
  Ogre* fiona = new Ogre();
                                     // Line 3
  shrek = *fiona;
                                     // Line 4
  delete fiona;
  return EXIT SUCCESS;
                                     // Line 5
}
```

(a) (12 points) Below, draw a diagram showing the contents of memory immediately after Line 3 is executed. Be sure to carefully distinguish between local (stack) variables and heap-allocated memory. Variables that are local to main should be drawn in a box labeled main on the stack.



(b) (4 points) Does this program execute successfully when it runs? If not, what goes wrong and why? (A couple of sentences should be more than enough.)

The program will crash with a duplicate delete error when the destructor for shrek attempts to delete its name_ array at the end of the program, because it will have been previously deleted in line 4 when the *fiona destructor was executed. The bug is that the default assignment operator leaves both Ogre objects pointing to the same array.

In addition, there is a memory leak because the assignment to shrek in line 3 overwrites its name_ pointer, which was the only thing referencing the first char array.

Question 8. (2 free points – all answers get the free points) Draw a picture of something you're planning to do this summer!



Congratulations on lots of great work this quarter! Have a great summer and best wishes for the future!! The CSE 333 staff