CSE 333 22au Final Exam 12/14/22

Name ____ UW netid ____________

There are 9 questions worth a total of 115 points. Please budget your time so you get to all of the questions. Keep your answers brief and to the point.

The exam is closed book, closed notes, closed electronics, closed telepathy, open mind. However, you may have two 5x8 notecards or the equivalent with any hand-written notes you wish written on both sides.

There are blank pages at the end with extra space for your answers if you need more room. It is after all the questions but before the detachable pages with reference information.

After the extra blank pages for answers, there are two pages of assorted reference information (much of which you probably won't need). You should remove those pages from the exam for convenience. These pages will not be scanned or graded.

Do not remove any pages from the middle of the exam.

If you do not remember the exact syntax for something, make the best attempt you can. We will make allowances when grading.

Don't be alarmed if there seems to be more space than is needed for some answers – we tried to include more than enough blank space.

Relax, you are here to learn.

Please wait to turn the page until everyone is told to begin

Score __________________ / 115

1. ______ / 20

2. ______ / 20

3. ______ / 20

4. ______ / 12

5. ______ / 15

6. ______ / 18

7. ______ / 4

8. ______ / 4

9. ______ / 2

Question 1. (20 points) A little C++/STL programming. It's the time of year when many packages are being sent across the country and around the world. We'd like to help our friends at the corner grocery-post-office-minimart-cafe keep track of the destinations of the many packages they are collecting that need to be sent somewhere. Every time somebody shows up with new packages to be shipped, our friends at the store add a line to the file `boxes.txt` listing the destination and number of new packages for that destination. A sample input file might look something like this:

```
Seattle    3
Redmond    1
Seattle    1
Omaha      4
Boston     3
```

(You should assume that each destination is a single word like "Seattle" or "Los_Angeles", not words with embedded blanks like "San Francisco".)

Write a small C++ program that accepts the name of a file formatted above as its argument (`argv[1]`), opens the file, and reads its contents. Then, after reading the file, the program should print a sorted list of the destinations and total number of packages being sent to each destination. For the above sample input file, the output would be

```
Boston     3
Omaha      4
Redmond    1
Seattle    4
```

(Do not worry about the exact spacing of the output as long as each line contains the proper information.)

You do need to check that the file can be opened successfully, but, to keep the code simple for this exam problem, you can assume that when you read the destination strings and integer numbers using the simple C++ >> stream input operator that it will work, and you do not need to check for whether that succeeds or not.

You should assume that all necessary headers have already been `#include`d (i.e., you do not need to worry about header files), and you can also assume that a `using namespace std;` directive has been written at the top of the file. It's also fine to write the whole program as a single `main` function if that makes sense, because it should be pretty short.

Hint: STL containers, particularly maps, might be very useful here….

Hint: remember that `ifstream f(`*filename*`, ifstream::in)` can be used to open the named file for reading, and you can test the stream variable `f` afterwards to see if the open succeeded by writing things like `if(f)...` or `if(!f)....`

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

Write your answer on the next page…

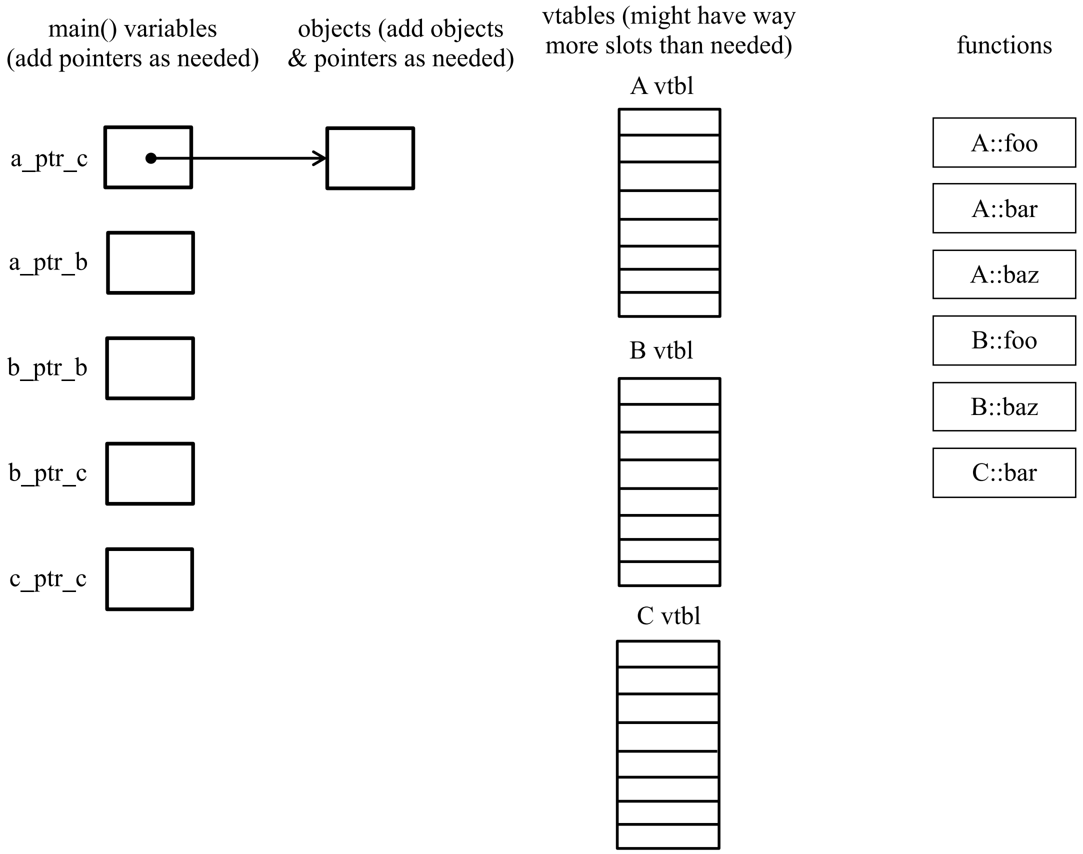
Question 1. (cont.) Write your C++ code below.

Question 2. (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:
  virtual void foo() = 0;
  virtual void bar() { baz(); cout << "A::bar" << endl; }
          void baz() { cout << "A::baz" << endl; }
};

class B : public A {
public:
  void foo() { bar(); cout << "B::foo" << endl; }
  void baz() { cout << "B::baz" << endl; }
};

class C : public B {
public:
  void bar() { cout << "C::bar" << endl; }
};

int main() {
  A* a_ptr_c = new C();
  A* a_ptr_b = new B();
  B* b_ptr_b = new B();
  B* b_ptr_c = new C();
  C* c_ptr_c = new C();

  a_ptr_b->foo();
  b_ptr_b->baz();
  c_ptr_c->bar();

  cout << "---" << endl;

  a_ptr_c->bar();
  c_ptr_c->baz();
  b_ptr_c->bar();

  cout << "---" << endl;

  c_ptr_c->foo();
  b_ptr_b->foo();
  a_ptr_c->foo();
  return EXIT_SUCCESS;
}
```

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

Question 2. (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.



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

Question 3. (20 points) Memories…. of CSE 333 and other things. A colleague is trying to build a small C++ class that holds an array of integers plus the length of the array. Here is the code they've created so far.

```
//// Ray.h ////
#ifndef RAY_H_
#define RAY_H_

#include <iostream>

class Ray {
 public:
  // construct new Ray with n copies of val
  Ray(int n, int val): size_(n), a_(new int[n]) {
    for (int i = 0; i < size_; i++) {
      a_[i] = val;
    }
  }
  // copy constructor and destructor
  Ray(const Ray &other) : size_(other.size_), a_(other.a_) { }
  ~Ray() { delete [] a_; }

  // print contents to stdout
  void Pr() {
    std::cout << "(";
    for (int k = 0; k < size_; k++) {
      std::cout << ((k > 0) ? "," : "") << a_[k];
    }
    std::cout << ")" << std::endl;
  }
 private:
  int size_;
  int *a_;  // elements stored in a_[0..size-1]
};
#endif // RAY_H_

//// main.cc ////
#include <cstdlib>
#include "Ray.h"

int main() {
  Ray r(2,17);
  Ray *p = new Ray(3,42);
  Ray s(r);
  // draw memory when execution reaches here //
  r.Pr();
  s.Pr();
  p->Pr();
  delete p;
  return EXIT_SUCCESS;
}
```

(continued on next page)

Question 3. (cont.) (a) (12 points) This program does compile and executes successfully, at least for a while. Draw a diagram showing the contents of memory when execution reaches the line in the `main` function with the comment that says “draw memory when execution reaches **here**”. Your diagram should clearly separate data on the stack from data allocated on the heap. Be sure to indicate clearly which data on the stack is part of the stack frame for function `main` by drawing a box labeled “main” with appropriate variables and data inside.

stack

heap

(continued on next page)

Question 3. (cont.) (b) (4 points) There is some sort of error in this program and it does not finish execution successfully. What, precisely, is the error and what goes wrong?

(c) (4 points) How can we fix the error that you described above in part (b) that causes the program to crash? You should describe what to change in the code and give the specific replacement C++ code needed to fix the problem.

Question 4. (12 points) Templates. The code from the previous question holds an array of `int`s and keeps track of the length of the array. Obviously, something like this could be useful for arrays of any type, not just `int`s.

Write your changes in the code below to modify the class and the main program that uses it so the class has a generic type parameter `E` for the type of the array elements, and the main program uses that template to create instances of the class whose elements are type `int`. The resulting program should work exactly the same as the original program, and will contain exactly the same bugs – don't fix those – but it should have a generic type for the array elements.

```
// Ray.h //
#ifndef RAY_H_
#define RAY_H_
#include <iostream>
class Ray {
 public:
  // construct new Ray with n copies of val
  Ray(int n, int val): size_(n), a_(new int[n]) {
    for (int i = 0; i < size_; i++) {
      a_[i] = val;
    }
  }
  // copy constructor and destructor
  Ray(const Ray &other) : size_(other.size_), a_(other.a_) { }
  ~Ray() { delete [] a_; }

  // print contents to stdout
  void Pr() {
    std::cout << "(";
    for (int k = 0; k < size_; k++) {
      std::cout << ((k > 0) ? "," : "") << a_[k];
    }
    std::cout << ")" << std::endl;
  }
```

(remainder of `Ray.h` and other code continued on next page)

Question 4. (cont.) Write the template changes needed on the rest of the code below.

```
//// remainder of Ray.h ////
 private:
  int size_;
  int *a_;  // elements stored in a[0..size-1]
};

#endif // RAY_H_

//// main.cc ////
#include <cstdlib>
#include "Ray.h"
int main() {
  Ray r(2,17);
  Ray *p = new Ray(3,42);
  Ray s(r);
  r.Pr();
  s.Pr();
  p->Pr();
  delete p;
  return EXIT_SUCCESS;
}
```

Question 5. (15 points) Networking. One of our friends is teaching themselves network programming and is using CSE 333 examples to learn how to build a network server. Unfortunately, they are having some problems and their code doesn't quite work right. The core parts of the code are shown below.

Write in corrections below to show how the code should be rearranged, changed, or fixed to create a properly working server. Feel free to draw arrows showing how to move code around if needed, but be sure it is clear to the reader what you mean.

You should assume that all functions always succeed – ignore error handling for this question. Further, assume that the `hints` data structure is set up correctly for the call to `getaddrinfo` and 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 "…" or missing 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));
  // Set up the hints...
  hints.ai_... = ...;    // specify values for options

  // Get local socket and create it
  getaddrinfo(NULL, argv[1], &hints, &rp); // assume *rp works
  int fd_1 = 0;
  socket(rp->ai_family, rp->ai_socktype, rp->ai_protocol);

  // Bind socket and set to listen
  int optval = 1;
  int sock_fam = rp->ai_family;
  setsockopt(listen_fd, SOL_SOCKET, SO_REUSEADDR,
                                   &optval, sizeof(optval));
  listen(fd_1, SOMAXCONN);
  bind(fd_1, rp->ai_addr, rp->ai_addrlen);
```

(code continued on next page)

Question 5. (cont.) Continue writing in your corrections below.

```
  // Accept clients and interact with them
  while (1) {
    int fd_2 = accept(fd_1, ...);
    // talk to client as needed
      read(fd_1);
      write(fd_1);
    close(fd_1);
  }
  return EXIT_SUCCESS;
}
```

Question 6. (18 points) Too many things at once. Consider the following small program that uses pthreads. (This does compile and execute successfully.)

```
#include <stdio.h>
#include <pthread.h>

int x = 0;
int y = 0;

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

int main() {
  pthread_t t1, t2;
  int ignore;
  ignore = pthread_create(&t1, NULL, &worker, NULL);
  ignore = pthread_create(&t2, NULL, &worker, NULL);
  pthread_join(t1, NULL);
  pthread_join(t2, NULL);
  printf("final x = %d, y = %d\n", x, y);
  return 0;
}
```

When we run this program, it starts two threads that each assign values to x and y and prints the values of those variables, then waits for both threads to finish, and then prints the final values of the variables x and y.

(a) (4 points) What output would this program print if the threads are executed sequentially, not concurrently? In other words, what output is produced if thread 1 executes first and then thread 2 executes after thread 1 terminates?

(continued on next page)

Question 6. (cont.) (b) (8 points) When the threads run concurrently, is it possible to get different output when the program is executed repeatedly? If it is, give three possible outputs that could be produced by the program. If there are only one or two possible outputs, write those and indicate that they are the only possible results. If one of the outputs is the same as in your answer to part (a) when the program runs sequentially, you can include that answer again here.

(You should assume that the statements in each individual thread are executed in the order written, and not rearranged by the compiler or memory system to be executed out-of-order. You should also assume that the `printf` calls don't interfere with each other and that each line of output is printed correctly on a single line and separately from other output lines, and the variables being printed are read simultaneously with no other thread interrupting to change them while `printf` is running. If different executions lead to different outputs it is only because of the interaction between statements in the threads the threads as they run concurrently.)

(c) (6 points) Assuming that the threads are executed concurrently, as in part (b), what are the possible final values of variables x and y? Circle all that could possibly happen on some possible execution:

Possible final values for x:

0 1 2 3 4 5 6 7 8 9 10 or more

Possible final values for y:

0 1 2 3 4 5 6 7 8 9 10 or more

A couple of short-answer questions to finish up. Keep your answers brief and to the point. It should be possible to answer these questions with a few sentences each.

Question 7. (4 points) The C++ standard libraries include smart pointers which automatically delete owned ("pointed-to") objects when the smart pointer is deleted. If used extensively these can help avoid all kinds of memory management problems, particularly memory leaks and dangling pointers. Could we use C++ smart pointers to automate all of our memory management, the same as is done by the automatic garbage collector in Java? Give short technical explanation of why or why not.

Question 8. (4 points) When we were exploring strategies for implementing concurrent servers, we looked at the relative costs of concurrent threads compared to processes. On most systems, processes seem to be at least an order of magnitude more expensive to create than threads. Why is this so? What is it about creating a process that is significantly more expensive than creating a new thread? (Be brief!)

Question 9. (2 free points – all answers get the free points) Draw a picture of something you're planning to do over winter break!

Congratulations on lots of great work this quarter!!
Have a great holiday break and best wishes for the new year!
The CSE 333 staff

Extra space for answers, if needed. Please be sure to label which question(s) are answered here, and be sure to put a note on the question page so the grader will know to look here.

Extra space for answers, if needed. Please be sure to label which question(s) are answered here, and be sure to put a note on the question page so the grader will know to look here.

Reference information. Here is a collection of information that might, or might not, be useful while taking the test. You can remove this page from the exam if you wish.

C++ strings: If `s` is a string, `s.length()` and `s.size()` return the number of characters in it. `s.find(`*search_string, start_pos* = `0)` returns the location of the first occurrence of *search_string* starting no earlier than *start_pos* or returns `string::npos` if not found. Subscripts (`s[i]`) can be used to access individual characters.

C++ STL:

- If `lst` is a STL `vector`, then `lst.begin()` and `lst.end()` return iterator values of type `vector<...>::iterator`. STL `list`s and `set`s are similar.
- A STL `map` is a collection of `pair` objects. If p is a `pair`, then `p.first` and `p.second` denote its two components. If the `pair` is stored in a `map`, then `p.first` is the key and `p.second` is the associated value.
- If `m` is a `map`, `m.begin()` and `m.end()` return iterator values. For a `map`, these iterators refer to the `Pair` objects in the `map`.
- If `it` is an iterator, then `*it` can be used to reference the item it currently points to, and `++it` will advance `it` to the next item, if any.
- Some useful operations on STL containers (lists, maps, sets, etc.):
  - `c.clear()` – remove all elements from c
  - `c.size()` – return number of elements in c
  - `c.empty()` – true if number of elements in c is 0, otherwise false
- Additional operations on `vector`s:
  - `c.push_back(x)` – copy x to end of c
- Some additional operations on `map`s:
  - `m.insert(x)` – add copy of `x` to `m` (a key-value pair for a `map`)
  - `m.count(x)` – number of elements with key `x` in `m` (0 or 1)
  - `m.find(`*item*`)` – iterator pointing to element with key that matches *item* if found, or `m.end()` if not found.
  - `m[k]` can be used to access the value associated with key `k`. If `m[k]` is read and has never been accessed before, then a <key,value> `Pair` is added to the map with `k` as the key and with a value created by the default constructor for the value type (0 or `nullptr` for primitive types).
- Some additional operations on `sets`
  - `s.insert(x)` – add `x` to `s` if not already present
  - `s.count(x)` – number of copies of `x` in `s` (0 or 1)
- You may use the C++11 `auto` keyword, C++11-style `for`-loops for iterating through containers, and any other features of standard C++11, but you are not required to do so.

More reference information. You can also remove this page if you wish.

Some POSIX I/O and TCP/IP functions:

- int accept(int sockfd, struct socckaddr *addr, socklen_t *addrlen);
- int bind(int sockfd, const struct sockaddr *addr, socklen_t addrlen)
- int close(int fd)
- int connect(int sockfd, const struct sockaddr *addr, socklen_t addrlen);
- int freeaddrinfo(struct addrinfo *res)
- int getaddrinfo(const char *hostname, const char *service,
  const struct addrinfo *hints, struct addrinfo **res)
  - Use NULL or listening port number for second argument
- int listen(int sockfd, int backlog)
  - Use SOMAXCONN for backlog
- off_t lseek(int fd, off_t offset, int whence)
  - whence is one of SEEK_SET, SEEK_CUR, SEEK_END
- ssize_t read(int fd, void *buf, size_t count)
  - if result is -1, errno could contain EINTR, EAGAIN, or other codes
- int socket(int domain, int type, int protocol)
  - Use SOCK_STREAM for type (TCP), 0 for protocol, get domain from address info struct (address info struct didn't fit on this page – we'll include it later if needed)
- ssize_t write(int fd, const void *buf, size_t count)

Some pthread functions:

- pthread_create(thread, attr, start_routine, arg)
- pthread_exit(status)
- pthread_join(thread, value_ptr)
- pthread_cancel (thread)
- pthread_mutex_init(pthread_mutex_t * mutex, attr) // attr=NULL usually
- pthread_mutex_lock(pthread_mutex_t * mutex)
- pthread_mutex_unlock(pthread_mutex_t * mutex)
- pthread_mutex_destroy(pthread_mutex_t * mutex)

Basic C memory management functions:

- void * malloc(size_t size)
- void free(void *ptr)
- void * calloc(size_t number, size_t size)
- void * realloc(void *ptr, size_t size)