

CSE 333 Final Exam August 21, 2015 **Sample Solution**

Question 1. (20 points) A bit of C++ coding. We've discovered an old C program that keeps track of student grades. It has been partially converted from C to C++, but the data is still held in a collection of `structs`. As in C, all of the fields of a C++ `struct` are public by default, but a C++ `struct` defines a regular, new type, not just a `struct` tag. The code uses some C++ types (`string`, `vector`). Otherwise this is still procedural code, not classes with member functions. Here are the declarations:

```
struct Course {           // information about one course
    string dept_;         // department ("CSE", "Art", ...)
    int num_;             // course number
    int credits_;        // credit hours
    float grade_;        // grade received (e.g., 4.0, 3.6, ...)
};

struct Transcript {       // a student transcript
    string name_;         // student name
    string major_;        // student department
    vector<Course> all_courses_; // courses taken
};
```

For this problem, implement function `GPAInDept` on the next page. This function has two arguments: a `Transcript` and a `string` giving a department name. The function should return the grade point average (`gpa`) for all courses found in the transcript from that department. The `gpa` is computed by selecting the courses whose department matches the requested one, multiplying each grade by the credit hours for the course, adding those results, then dividing by the total credit hours for all selected courses.

Example: suppose the transcript contains: CSE 143, 5 credits, grade 3.6; History 100, 3 credits, 2.8; CSE 333, 4 credits, 3.8; and Art 490, 3 credits, 4.0. The `gpa` for "CSE" is $(3.6*5 + 3.8*4) / 9 = 3.69$.

Some brief reference information about the STL `vector` class, used in `Transcript`.

- If `lst` is a STL `vector`, then `lst.begin()` and `lst.end()` return iterator values of type `vector<...>::iterator` that might be useful.
- 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, including `vector`:
 - `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
 - `c.push_back(x)` – copy `x` to end of `c`
- You are free to 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 use these.

Write your code on the next page. You can remove this page for reference if you wish.

CSE 333 Final Exam August 21, 2015 **Sample Solution**

Question 1. (cont.) Write your implementation below. The function heading is given for you, and you should assume that all necessary headers have already been #included. Hint: you probably won't need nearly all of this space.

```
// Return gpa for all courses in Transcript t where the
// course department matches the parameter dept.
// Return 0.0 if t contains no courses that match dept.
float GPAInDept(const Transcript &t, const string &dept) {
    int total_credits = 0;
    float total_points = 0.0;
    for (auto &course: t.all_courses_) {
        if (dept == course.dept_) {
            total_credits += course.credits_;
            total_points += course.credits_ * course.grade_;
        }
    }
    if (total_credits == 0)
        return 0.0;
    else
        return total_points / total_credits;
}
```

Second solution using explicit iterators to process the list of courses:

```
float GPAInDept(const Transcript &t, const string &dept) {
    int total_credits = 0;
    float total_points = 0.0;
    // for (auto course: t.all_courses_) {
    for (auto it = t.all_courses_.begin();
         it < t.all_courses_.end(); ++it) {
        if (dept == it->dept_) {
            total_credits += it->credits_;
            total_points += it->credits_ * it->grade_;
        }
    }
    if (total_credits == 0)
        return 0.0;
    else
        return total_points / total_credits;
}
```

CSE 333 Final Exam August 21, 2015 **Sample Solution**

Question 2. (16 points) Smart pointers. The following program creates a short linked list. But it leaks memory because it never deletes any of the heap-allocated data.

Fix this program so it has no memory leaks. However, you may not alter what the program does, you may not replace pointers to data with copies of the data, and you may not insert any `delete` statements. Instead, you should fix the leaks by changing the code to use smart pointers appropriately instead of regular pointers. Cross out existing code and write new code as needed. Legibility is a big help – please write clearly.

Changes needed shown in bold below.

```
#include <iostream>      // needed only for printing list
#include <memory>        // smart pointers (for solution)
using namespace std;

struct Node {
    unique_ptr<int> val_; // ptr to node's data on heap
    shared_ptr<Node> next_; // next node in list or nullptr if none
};

int main() {
    // create list
    shared_ptr<Node> list( new Node() );
    list->val_ = unique_ptr<int>( new int(17) );
    shared_ptr<Node> p( new Node() );
    p->val_ = unique_ptr<int>( new int(42) );
    p->next_ = nullptr;
    list->next_ = p;
    // print list
    for (auto n = list; n != nullptr; n = n->next_)
        cout << *(n->val_) << " ";
    cout << endl;
    return 0;
}
```

Notes: The payload (`val_`) pointer could have also been a `shared_ptr`, but since each node has the only pointer to its data a `unique_ptr` is more efficient. Pointers to the list nodes need to be `shared_ptr`s, however, to allow for assignments.

There are other ways to solve the problem and correct solutions received credit provided they only modified the program to use smart pointers and did not change it otherwise.

CSE 333 Final Exam August 21, 2015 **Sample Solution**

Question 3. (16 points) Not the inheritance question you were expecting. Give C++ code that could be put in place of the `/* YOUR CODE HERE */` below such that the program compiles without warning and prints out the 13 characters `Hello, World!` followed by a newline (and nothing else) when run.

- You must provide one class definition and two method *definitions* (you can *declare* more methods.)
- Sample solution is 8 lines (but, like the supplied code, some methods are defined in one line). Your solution doesn't need to have exactly the same number of lines.

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

/* YOUR CODE HERE */

class D : public C {
public:
    void m1();
    void m2();
};
void D::m1() { cout << "summer "; }
void D::m2() { cout << ", World"; }

int main(int argc, char** argv) {
    C* c = new D();
    char x = 'F';
    c->m1();
    c->m2();
    c->m3(x);
    cout << x << endl;
    return 0;
}
```

Solution:

```
class C {
public:
    void m1();
    virtual void m2() = 0;
    void m3(char& a);
};
void C::m1() { cout << "Hello"; }
void C::m3(char& x) { x = '!'; }
```

Note: `m2` must be a pure virtual (abstract) function in order to meet the problem constraint of only two method definitions in the solution.

CSE 333 Final Exam August 21, 2015 **Sample Solution**

Question 4. (15 points) Memory mangling. Suppose we have the following struct to define nodes of a linked list whose data consists of heap-allocated C strings:

```
struct node {           // heap-allocated node
    char *s;           // pointer to string on the heap
    struct node *next; // pointer to next node, or NULL if none
};
```

Here are three functions that attempt to free one of these linked lists. For each one, write “WORKS” if the function works properly and frees all of the list and data, write “LEAK” if the function leaks memory (fails to free something), or write “DANGLE” if the function uses a dangling pointer (i.e., references memory using a pointer after that memory is freed). If there is a problem, give a brief description of what’s wrong – you don’t need to fix it. If there is more than one problem, identify all of them.

(a) (5 points)

```
void free_list_1(struct node *lst) {
    if (lst == NULL)
        return;
    free(lst);
}
```

LEAK: frees only the first node, but none of the other nodes or strings.

(b) (5 points)

```
void free_list_2(struct node *lst) {
    if (lst == NULL)
        return;
    free(lst->s);
    free_list_2(lst->next);
    free(lst);
}
```

WORKS

(c) (5 points)

```
void free_list_3(struct node *lst) {
    if (lst == NULL)
        return;
    free(lst);
    free(lst->s);
    free_list_3(lst->next);
}
```

DANGLE: references pointers in the first node after that node has been freed.

CSE 333 Final Exam August 21, 2015 **Sample Solution**

Question 5. (15 points) The list below describes several of the tasks that are performed by various TCP/IP networking functions. For each task, indicate which TCP/IP functions perform that task by writing the function name or names below the task description. A list of the possible functions is given at the end of the question. Note that some tasks are performed by more than one function, but all of them are performed by at least one function. Be sure your answers are specific – list only the function or functions that perform the immediate task described. Don't list other functions that might have been called previously, even if these would be needed before doing the particular task.

- a) Allocate a new file descriptor entry in the table of active file descriptors for this process.

socket, accept

- b) Open a TCP stream so that it can be used to transmit bytes to or receive bytes from the stream.

connect, accept

- c) Exchange (send or receive) bytes between a client and a server in either direction.

read, write

- d) Translate a domain name like “attu.cs.washington.edu” to an IP address like 128.208.1.138.

getaddrinfo

- e) In a server program, indicate to the operating system that it should allow clients to request service and that any requests received should be queued up until the server is ready to handle them.

listen

List of possible networking functions:

| | | |
|-----------|---------------|---------------|
| accept() | inet_ntop() | getnameinfo() |
| bind() | inet_pton() | read() |
| close() | listen() | socket() |
| connect() | getaddrinfo() | write() |

CSE 333 Final Exam August 21, 2015 **Sample Solution**

Question 6. (16 points) Concurrency. Consider the following program (`#includes` omitted to save space). The program calls function `do_work` many times to process parts of some larger job. The exact work done doesn't matter and isn't specified further, except that each call to `do_work` is completely independent of the rest of the code and does not share any data with anything else. One of the interns has added threads to try to take advantage of a multi-core system to get the total work done faster. There is a global variable `work_done` that records the number of times `do_work` has been called so far.

```
const int NUM_THREADS = 4;    // # worker threads
const int NUM_ITER = 10000;  // # steps to be done by each worker

static int work_done = 0;
static pthread_mutex_t sum_lock;

// Do one chunk of work
void do_work() { ... details omitted ... }

void *thread_main(void *arg) {
    pthread_mutex_lock(&sum_lock);
    for (int i = 0; i < NUM_ITER; i++) {
        do_work();
        work_done++;
    }
    pthread_mutex_unlock(&sum_lock);
    return NULL;
}

int main(int argc, char** argv) {
    pthread_t thds[NUM_THREADS];
    pthread_mutex_init(&sum_lock, NULL);

    for (int i = 0; i < NUM_THREADS; i++) {
        if (pthread_create(&thds[i], NULL, &thread_main, NULL) != 0) {
            std::cerr << "pthread_create failed" << std::endl;
            exit(1);
        }
    }

    std::cout << "Total: " << work_done << std::endl;
    pthread_mutex_destroy(&sum_lock);
    return 0;
}
```

Answer questions about this code on the next page. You may remove this page from the exam if that is convenient.

CSE 333 Final Exam August 21, 2015 **Sample Solution**

Question 6. (cont.) (a) (6 points) When testing this code, it seems to run more slowly than expected – not much faster than the original single-threaded version. What’s the bug in the code that causes this particular problem and how should it be fixed?

The lock in `thread_main` is held too long. It should be acquired right before incrementing `work_done` and released immediately after so other threads can execute while `do_work()` is running.

(b) (5 points) The `main` function prints the value of variable `work_done` after it creates all the threads. If we execute the original program with no changes, which of the following values could be printed by that output statement? Circle all of the ones that are possible. Note that there might be additional values that could be printed that are not listed. Just indicate from the ones given here which ones could appear.

| | | | |
|------|-------|-------|-------|
| 0 | 8586 | 20000 | 36157 |
| 1 | 10000 | 25305 | 40000 |
| 1571 | 11110 | 30000 | 42000 |

(c) (5 points) If the program can print any value other than 40000 (i.e., number of threads times number of iterations per thread), what is the bug that allows this to happen, and how should it be fixed? If the program always produces the right answer and there is nothing to be fixed, say so. If a fix is needed, your solution should still allow concurrent execution of the threads (i.e., “remove concurrency” is not the right answer). You do not need to provide detailed code; just give a specific explanation of what needs to be done.

The main program prints `work_done` and terminates immediately after calling `pthread_create` to create the four threads. Depending on how the threads are scheduled, all of the new threads might run to completion before `main` exits, or the process could terminate immediately before any of the threads have done any work at all.

The fix is for the main thread to call `pthread_join` or some other appropriate function(s) to wait for the four workers to finish before printing the final value of `work_done` and then terminating the program.

Question 7. (2(!) free points) We should have had a clever or funny (or at least mildly amusing) question here, but couldn’t come up with anything. What should it have been?

All answers received both points.