There are 7 questions worth a total of 100 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.

If you don’t 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 your 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 __________________ / 100

1. _____ / 20
2. _____ / 18
3. _____ / 15
4. _____ / 15
5. _____ / 16
6. _____ / 15
7. _____ / 1
Question 1. (20 points) A bit of C++ hacking – STL version. The C++ 11 standard library contains a container unordered_set<T> that stores sets of elements with no duplicates. Internally the set elements are stored using a hash table, so there is no guarantee about the order in which the elements are retrieved by an iterator.

On the next page, give the full implementation of a function intersect that returns a new unordered_set that is the intersection of its two arguments. For example, if we execute the following code:

```cpp
unordered_set<int> s1 { 22, -5, 333, 17, -12, 42 };  
unordered_set<int> s2 { 42, 5, 333, 9, 17, 0 };  
unordered_set<int> result = intersect(s1, s2);
```

the variable result should contain the unordered collection of values { 17, 333, 42 }, i.e., only values that appear in both of the argument sets. The intersect function must not make any changes to either of its arguments. Your function will need one unordered_set variable to accumulate the result, but it may not use any other STL containers, C++ arrays, or other complex data structures.

Your solution only needs to work with argument and result values of type unordered_set<int>, i.e., sets of integers. You need to provide proper declarations for the parameter and result types for the function, as well as the code to implement it. You may, if you wish, use a function template so it will work with any element type, but there is no extra credit for doing so. Your code should be written in good style.

Some reference information: unordered_set has the same interface as set, except it does not guarantee the order in which values are stored or are retrieved by an iterator.

- The default (0 argument) constructor creates an empty unordered_set. Copy constructors and initializer lists can also be used to create new sets.
- As with any STL container, if s is a unordered_set, s.begin() and s.end() return iterator values 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 all STL containers, including unordered_set:
  - 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
- Some additional operations on unordered_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 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 answer on the next page. You may remove this page for reference while working if you wish.
Question 1. (cont.) Write your implementation of function intersect below.
Question 2. (18 points) C++ programming – with templates this time. Write a C++ template for a function `max` that has two parameters, an array `a` and an integer `n`. Function `max` should return the largest value found in the first `n` elements of `a`. For example, if we execute the following statements,

```cpp
int ints[5] = { 22, -5, 333, 17, 42 };
cout << "largest int is " << max(ints, 5) << endl;
cout << "largest string is " << max(strings, 3) << endl;
```

the output should be (note that only 3 elements of the string array are examined):

```
largest int is 333
largest string is harpo
```

Your code should work on any array type whose elements can be compared with ‘<’. You may not use any additional arrays, STL containers, or other complex data structures. You should assume `n>=1`, i.e., the array has at least one element. Use good style.
Question 3. (15 points) Dynamic dispatch. The following program prints howle rorwl. Add the keyword virtual in appropriate places so that the program will print hello world instead. You may not make any other changes to the code.

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

class A {
public:
    void m1() { cout << "h"; }
    void m2() { cout << "o"; }
    void m3() { cout << "w"; }
};
class B : public A {
public:
    void m1() { cout << "r"; }
    void m2() { cout << "e"; }
    void m3() { cout << "l"; }
};
class C : public B {
public:
    void m1() { cout << "w"; }
    void m2() { cout << "o"; }
    void m3() { cout << "d"; }
};

int main() {
    A* ab = new B();
    B* b = new B();
    B* bc = new C();
    A* ac = new C();
    ac->m1();
    ab->m2();
    ab->m3();
    b->m3();
    bc->m2();
    cout <<" ";
    bc->m1();
    ac->m2();
    b->m1();
    ab->m3();
    bc->m3();
    cout << endl;
    return 0;
}
```
Question 4. (15 points) Pointy Things. Consider the following program:

```cpp
#include <memory>
#include <stdlib.h>
#include <iostream>
using namespace std;

class Thing {
private:
  int id_
public:
  Thing(int n): id_(n) {}
  ~Thing() { cout << "destruct " << id_ << endl; }
};

void DoingThings(shared_ptr<Thing> &p){
  Thing* b = new Thing(2);
  unique_ptr<Thing> c(new Thing(3));
  c.reset(new Thing(4));
  cout << "new scope" << endl;
  {
  unique_ptr<Thing> d(new Thing(5));
  shared_ptr<Thing> e = p;
  }
  cout << "got here" << endl;
}

int main(){
  shared_ptr<Thing> a(new Thing(1));
  DoingThings(a);
  unique_ptr<Thing> g(new Thing(6));
  cout << "another scope" << endl;
  {
  shared_ptr<Thing> j = a;
  Thing* i = g.release();
  delete i;
  }
  cout << "almost done" << endl;
}
```

Answer questions about this program on the next page. You may detach this page for convenience if you wish.
Question 4. (cont.) (a) (11 points) What output does the above program produce when it is executed?

(b) (4 points) Are there any memory leaks, dangling pointers, or other memory management errors in this program? If so, describe the errors and where they are in the code. (You can give a brief verbal description; you don’t need to annotate the code unless you want to.) If a Thing object is leaked, include its id_ number in your description; if there are dangling pointers or other problems, identify the specific pointer(s) and describe the problem(s) – briefly!
Question 5. (16 points) Games. Here is a sketch of a program that plays a game with a user using standard console input and output. The details of the game don’t matter for this question. We assume that I/O operations are done with POSIX read/write functions called from inside the `GetUserMove` and `WriteComputerMove` functions.

```c
// Read the next user move from the given input file descriptor
// and return a numeric code describing the move. 0 means quit;
// non-zero is a specific move.
int GetUserMove(int input_file_descriptor) { ... }

// Given a code describing the last user input move, compute a
// response and write it to the given output file descriptor
void WriteComputerMove(int user_input,
                        int output_file_descriptor) { ... }

// play a game using standard console I/O until the user quits.
const int STDIN_FD = 0;
const int STDOUT_FD = 1;

int main() {
    while (1) {
        int user_move = GetUserMove(STDIN_FD);
        if (user_move == 0)
            break;  // game over
        WriteComputerMove(user_move, STDOUT_FD);
    }
    return 0;
}
```

We would like adapt this code to create a “game server” to play this game with a network user using the same `GetUserMove` and `WriteComputerMove` functions and basic game loop shown above. You need to write server pseudo-code to create a connection to a client machine, read input from the client and write answers back, and close the connection and terminate when the user ends the game (i.e., when `GetUserMove` returns 0). The server only needs to handle one client and should shut down when the game with that single client is done – i.e., you don’t need to handle multiple clients.

The pseudo-code should have a similar level of detail as the code above, but it should be clear where sockets are set up, which file descriptors are used at various points in the code, and so forth. But you don’t need to worry about all of the grubby details of the exact socket parameter lists. The next page summarizes many networking functions, and you will want to use several of them in your answer.
Question 5. (cont.) Reference information:

Here is a sketch of some TCP, IP, and POSIX I/O functions and data structures that may be useful in your solution. You do not need to provide detailed values for all arguments as you would in a real program, but you do need to keep track of file descriptors and sketch when address information is created and used, so it is clear how connections are set up and which descriptors are used for input and output. You should assume that all functions succeed on the very first try and you do not need to check for errors.

You will not need to use all of these functions in your answer.

If it matters, you can use port 1337 for your server, but any number is fine.

- struct addrinfo – address data structure; appears as address_info in several function descriptions, below
- int accept(file_descriptor, address_info),   // returns a file_descriptor
- bind(file_descriptor, address_info)
- close(file_descriptor)
- connect(file_descriptor, address_info);
- freeaddrinfo(struct addrinfo *result)
- getaddrinfo(NULL, port, struct addrinfo *hints, struct addrinfo **result)
- listen(file_descriptor)
- int read(file_descriptor, buffer)   // returns # bytes read or error
- int socket(address_info)                  // returns a file_descriptor
- int write(file_descriptor, buffer)   // returns # bytes written or error

Write the pseudo-code for your game server on the next page. You can detach this page, too, for reference if you want.... 😊
Question 5. (cont.) Write your game server pseudo-code below.
Question 6. (15 points) Doing (too?) many things at once? The following C++ code is part of a class that implements a stack of integers using a linked list representation.

```cpp
1 class Stack {
2     public:
3         // construct new, empty stack
4         Stack() : top_(nullptr) { }
5
6         // push n onto top of stack
7         void push(int n) {
8             top_ = new Node(n, top_);
9         }
10
11         // return top value from stack and remove it
12         // pre: stack is not empty
13         int pop() {
14             int topval = top_->val_;    
15             Node *p = top_;           
16             top_ = top_->next_;       
17             delete p;               
18             return topval;          
19         }
20
21     private:
22         // stack representation is a linked list of Nodes
23         class Node {
24             public:
25                 int val_;     // integer value in this Node
26                 Node *next_;  // Node below this one, nullptr if none
27             // convenience constructor
28             Node(int value, Node* next): val_(value),next_(next) {} 
29         };
30
31         Node* top_;    // ptr to top Node; nullptr if empty stack
32     };
```

Other operations, including the destructor, are omitted to save space.

Answer questions about this class on the next page. You may want to leave this page in the exam to hand in with your answer to part (b).
Question 6. (cont.) (a) (8 points) This code is not safe to use in a multi-threaded, concurrent program. Give a concrete example that explains a problem that can occur if this code is used in a multi-threaded program. (There may be many possible examples, you just need to give one.)

(b) (7 points) Assume that we have a class Lock with the following operations:

```java
Lock lock; // construct a new Lock variable that is unlocked
lock.Acquire(); // delay thread until lock is available then acquire the lock
lock.Release(); // release (unlock) the lock. If other threads are waiting
                // for the lock to be available, one of the pending Acquire
                // operations will succeed and that thread will run
```

In the code for class Stack on the previous page, show where to insert one or more Lock variables and Acquire and Release operations so the Stack class can be used safely by concurrent, multi-threaded programs. Or you can describe the changes below, but be specific and be sure it is crystal clear what changes should be made and where.
Question 7. (1 free point) The very best telephone of all time is (circle):

a) Palm Treo
b) Nokia 9000
c) Motorola TAC
d) BlackBerry
e) iPhone
f) Nokia Windows phone
g) Samsung Galaxy
h) Motorola Android
i) Western Electric 500
j) Generic flip-phone
k) You are clueless! The correct answer is _________________________!!
l) I don’t care, but please give me my free point!
m) I do care, but I’m not telling. Give me my free point anyway!!
n) Other: