CS 3214 Final Exam

This is a closed-book, closed-internet, closed-cell phone and closed-computer exam. However, you may refer to your 2 sheets of prepared notes. **Your exam should have 12 pages with 5 topics totaling 150 points. You have 120 minutes**. Please write your answers in the space provided on the exam paper. If you unstaple your exam, please put your initials on all pages. You may use the back of pages if necessary, but please indicate if you do so we know where to look for your solution. Answers will be graded on correctness and clarity. You will lose points if your solution is more complicated than necessary or if you provide extraneous, but incorrect information along with a correct solution.
To be considerate to your fellow students, if you leave early, do so with the least amount of noise. Students who have not finished after 100 mins are asked to stay until the end to not disturb others.

Name (printed) ____

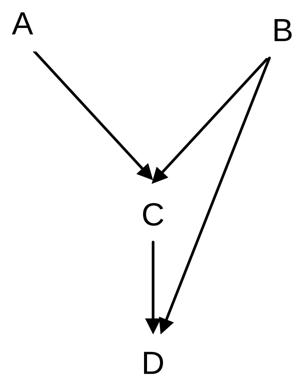
I accept the letter and the spirit of the Virginia Tech undergraduate honor code – I have not given or received aid on this exam.

(signed) ____

#	Problem	Points	Score
1	Multithreading	45	
2	Memory Organization and Addressing	30	
3	Dynamic Memory Management	25	
4	Internet Addressing and Protocols	30	
5	Essay Question: Parallel Programming and Education	20	
	Total	150	

1. Multithreading (30 pts)

a) (12 points) *Semaphores*. Semaphores can be used to express scheduling constraints between activities performed by different threads. Consider the figure shown on the right, denoting the dependencies between activities A, B, C, and D, which are executed by 4 different threads.


$$\begin{tikzcd} A \arrow[dr] & & B \arrow[dl] \arrow[ddl] \\ & C \arrow[d] & \\ & D & \end{tikzcd}$$

Complete the program below to ensure these constraints!

// declare any semaphores you need here;
// be sure to show how to initialize them
sem_t adone, bdone, cdone;
sem_init(&adone, 0, 0);
sem_init(&bdone, 0, 0);
sem_init(&cdone, 0, 0)’
	2 points declaration
	2 points initialization
	1 point correct number of semaphores

static void *
thread_A(void *_) {
 	
printf(“A\n”);
sem_post(&a_done); 1 point
 	
}
	static void *
	thread_B(void *_) {
	
	printf(“B\n”);
	sem_post(&b_done); 1 point
	
	}

static void *
thread_C(void *_) {
 	
sem_wait(&adone); 2 points
sem_wait(&bdone);
 	
printf(“C\n”);
 	
sem_post(&cdone); 1 point
 	
}
	static void *
	thread_D(void *_) {
	
	sem_wait(&bdone); 2 points
	sem_wait(&cdone);
	
	printf(“D\n”);
	
	}

```
int main()
{
    pthread_t t[N];
    pthread_create(t+0, NULL, thread_D, NULL);
    pthread_create(t+1, NULL, thread_C, NULL);
    pthread_create(t+2, NULL, thread_B, NULL);
    pthread_create(t+3, NULL, thread_A, NULL);
    pthread_exit(0);
}
```

b) (18 pts) *Condition Variables.* In scientific computing a *barrier* is often used to synchronize NTHREADS concurrent threads. Each thread arriving at the barrier is forced to wait until all NTHREADS threads have arrived at the barrier. Only when all NTHREADS threads have arrived and are all waiting at the barrier are they allowed to resume execution. Assume that NTHREADS is given a value via a #define.

i. (5 points) Show the definition of a struct type named barrier that you would use to implement barrier synchronization in Pthreads.

```
struct {
  pthread_mutex_t mutex;
  pthread_cond_t  halt;
  int count;
 } barrier;
```

2 points for including a condition variable
2 points for including a mutex variable
1 point for including a count

ii. (4 points). Show the code needed to implement the function to initialize an instance of your barrier structure. The signature of this function is:

barrier* barrier_create();

```
barrier* barrier_create() {
   barrier* bp = (barrier*)malloc(sizeof(barrier));
   pthread_mutex_init(bp->mutex, NULL);
   pthread_cond_init(bp->halt, NULL);
   bp->count = 0;
}
```

1 point for allocating the structure correctly
1 point for initializing the mutex
1 point for initializing the condition variable
1 point for initializing the count

iii. (9 points) Show the code needed to implement the function executed by threads when they arrive at a barrier. Assume that your barrier struct has been properly initialized. The signature of this function is:

void barrier_arrive(barrier *bp);

```
void barrier_arrive(barrier *bp) {
    pthread_mutex_lock(bp->mutex);
    bp->count++;
    if (bp->count < NTHREADS)
      pthread_cond_wait( bp->halt, bp->mutex);
   else {
     pthread_cond_broadcast(bp->halt);
     pthread_mutex_unlock(bp->mutex);
   }
}
```

2 points for locking the mutex to create a critical section
1 point for incrementing the count
3 points for testing the count and waiting if count < NTHREADS
3 points for broadcasting the signal and unlocking the semaphore

c) (9 points) When each of the following system call is performed by one thread explain what if any effect it has on other threads in the same process.

i. the read system call.

No effect. **2 points**

ii. The exit system call.

Terminates all threads. **3 points**

iii. The fork system call.

No effect. **2 points**

iv. The exec system call.

Terminates all threads. **2 points**

d) (6 points) Give two causes for a function to be considered thread unsafe?

it call a thread unsafe function
it returns a pointer to static memory
it keeps state across invocations
it does not protect shared variables

3 points for each correct cause given from among 4 listed above

2. Memory Organization and Addressing (20 points)

a) (12 points) A security application has *N* different face recognition algorithms that have approximately equal running times. It is presented with a large file of *M* images to be scanned. The images are of approximately the same size. Typically, *M* >> *N (i.e., M* is much bigger than *N)*. The application uses *mmap* to access in memory the image(s) it is scanning. The code for the face recognition algorithms are in a dynamically shared library.

On a multi-core system with kernel-supported threads, consider the following design alternatives (a)-(d) shown in the table below. The significant memory cost factor is the image data. Using order of magnitude (big-O) notation, determine the virtual and physical memory costs of the four alternatives above and enter them in the following table.

	Virtual Memory Size	**Physical Memory Size**
(a) *M* processes each applying the *N* algorithms to a different image	**O(M) 1 point**	**O(M) 1 point**
(b) *M* threads in one process each applying the *N* algorithms to a different image	**O(M) 1 point**	**O(M) 1 point**
(c) *N* processes each applying a different algorithm to all the images	**O(MxN) 2 points**	**O(N) 2 points**
(d) *N* threads in one process each applying a different algorithm to all the images	**O(M) 2 points**	**O(N) 2 points**

b) (18 points) A structure type *S* with a member *elem* is partially defined and referenced as follows:

```
struct {
  …
  int elem;
  …
} S;

struct S *sp;
struct S inst;
```

The space for the structure could be allocated in one of three ways:

global: sp = &inst; (i.e., pointing to a global program variable)
malloc: sp = (S*)malloc(sizeof(S));
mmap: sp = mmap(NULL, sizeof(S), …)

At runtime the reference *sp->elem* might appear as a machine instruction of this form*: mov offset(base), register* that loads into a register the value at an address in physical memory. The parts of this instruction are determined by some combination of actions taken by system components (compiler, linker/loader, run-time system, or operating system) which may different depending on how the structure is allocated.

List in each box of the table below the number of one or more of the following seven actions to show what role each of the identified components has on determining the physical address. For example, if you believe that the compiler has no role in the case of global allocation then enter a 7 in the upper left box.

Actions:
1. determines base address
2. determines offset value
3. determines both base address and offset value
4. determines cache line used
5. determines virtual address to physical address mapping
6. determines the register used
7. no role

	compiler	**linker/loader**	**run-time system**	**operating system**
global	**2,6**	**1**	**7**	**5**
malloc	**2,6**	**7**	**1**	**5**
mmap	**2,6**	**7**	**7**	**1,5**

1 point for each correct answer in above table
2 points discretionary

3. Dynamic Memory Management (15 points)

a) (10 points) As in project 4 assume that a dynamic memory allocator uses header and footer blocks (forming an implicit list) and a separate explicit free list is maintained where the pointers for the free blocks are kept in the free block itself. Assume the following programming notation:

   if *p* is a pointer to a header or footer then *p->length* give the length recorded in the header or footer and *p->status* is a Boolean variable indicating that the block is free (true) or allocated (false)

   the first element of the free list is pointed to by the variable *free* and the header of the first block is pointed to by the variable *first*

   if *elem* in a pointer to a element in the free list then *elem->addr* is the address of the start of the free block (i.e., the address the block's header), *elem->next* points to the next element and the function *done(elem)* indicates that the end of the list has been reached

   <u>Write approximately correct code for the function that return the total amount of free memory using the implicit free list. The signature of this function is:</u>

   int free_size(block * first);

```
int free_size(block *first) {

  block *p = first;
  int size = 0;
  while(! (p->status && p->length==0))
    if (p->status) size = size + p->length;
    p = p + p->length
}
```

2 points for correct initialization of starting pointer
4 points for correct condition on while loop (finding the end marker)
2 points for initializing the size counter and incrementing it
2 points for updating the pointer correctly to next element in free list

b) (8 points) An application uses a large (0.5 Gb-1.0 Gb) dynamically allocated memory. Like project 4, an explicit free list is maintained where the pointers for the free block are kept in the free block itself. The memory allocator is parameterized and allows the user to choose how the free list is organized (smallest block first, FIFO, address ordered) and what algorithm to use (first-fit, best-fit).

Given your understanding of the full range of memory management issues, what combination of parameters would you choose for the list organization and the algorithm if the goal was to minimize allocation time. Explain your choice.

The first-fit algorithm should be used.
The list should be ordered by addresses to avoid page faults that would occur using a randomly ordered list.

4 points for selecting the correct algorithm (2 points if they give the wrong answer but have some reasonable explanation)

4 points for giving the correct ordering and giving the correct reason (2 points for giving the correct ordering but not a good reason)

c) (7 points) Consider a Java class for a list whose methods allow elements to be added, searched for, and removed. Explain how this class could be a source of a form of "memory leaks" in Java.

If objects are added to the list and then never subsequently used (because they were not correctly removed from the list) then memory would continue to be allocated for an object that will not be used again.

4 points for describing how the list could contain items that would never be used outside of the list itself

3 points for describing how this is a memory leak (i.e., consumes memory that plays no useful role in the computation)

Note: grading more discretionary than in other questions. Feel free to award significant partial credit for answer demonstrating good understanding of issues involved.

4. Internet Addressing and Protocols (20 pts)

a) A user provides to a web browser this URL of an HTTP server: http://www.company.com/updates/recent.html

i. (8 points) In the table below write the specific parts of the above URL that are used for each purpose stated in the table. Write "none" if no part of the URL applies to the stated purpose.

Purpose	Part of URL or "none"
(a) determines server port number	**http** **2 points**
(b) determines client port number	**none** **1 point**
(c) determines server internet address	**www.company.com** **1 point**
(d) determines client internet address	**none** **1 point**
(e) is used by the Domain Name System	**www.company.com** **1 point**
(f) determines that a connection-oriented protocol is used	**http** **1 point**
(g) determines that a persistent connection is used	**none** **1 point**

ii. (8 points) What is the first HTTP command that the browser would send to the server using verion 1.1 of the HTTP protocol and indicating that a persistent connection should be used? Use the notation <CRLF> to show the presence of "\r\n" in the command.

GET /updates/recent.html HTTP/1.1<CRLF>
HOST: www.company.com <CRLF>
CONNECTION: keep-alive <CRLF>
<CRLF>

3 points for initial GET command
2 points for HOST header line
1 point for CONNECTION header line
1 points last <CRLF> line
1 point for ending each line with <CRLF>

b) (9 points) A NAT router with the internet address 128.119.40.186 is used to create a private local network. One of the machines on this private local network has the internet address 192.168.5.62. A client on this machine opens a connection to a server on a machine not in the private local network using, in part, this code:

```
int err = getaddrinfo("www.myserver.com", "3344", …)
```

The client's port is assigned the port number 1234 by the client's local host. Assume that the server www.myserver.cm is at internet address 128.173.41.81. An open socket is identified by a quadruple S: (source address, source port), D: (destination address, destination port). The next port number to be assigned by the NAT is 5678.

i. What is the quadruple for the socket on the client?

S: (192.168.5.62, 1234)
D: (128.173.41.81, 3344)

3 points for correct source and destination

ii. what is the quadruple for the socket on the server?

S: (128.173.41.81, 3344)
D: (128.119.40.186, 5678)

3 points for correct source and destination

iii. What is the relevant table entry in the NAT translation table?

(128.119.40.186, 5678) → (192.168.5.62, 1234)

3 points for correct table entry

c) (5 pts) Indicate which of the following statements are true for HTTP servers.

i. A stateless server may not use a persistent connection because that means it is maintaining state.

ii. A server typically uses a separate process to serve dynamic content.

iii. A single server may not have both persistent and non-persistent connections at the same time.

iv. If a client sends two GET requests over the same connection the server cannot begin processing the second request until it has finished processing the first request.

v. The same server process can handle requests for two different host names (e.g., www.company.com and www.business.com).

i. is false
ii. is true
iii. is false
iv. is true
v. is true

1 point for each correct answer

5. Essay Question: The NSF/IEEE TCPP Curriculum (20 points)

IEEE's Technical Committee on Parallel Processing (TCPP), supported by funding from the National Science Foundation (NSF) and Intel and IBM Corporations, has been developing a Parallel and Distributed Computing Curriculum that seeks to *embed parallel and distributed computing throughout all courses in an undergraduate Computer Science (CS) curriculum*. The authors of this proposal write:

"In the past, it was possible to relegate issues regarding parallelism and locality to advanced courses that treat subjects such as operating systems, databases, and high performance computing: the issues could safely be ignored in the first years of a computing curriculum. But it is clear that changes in architecture are driving advances in languages that necessitate new problem solving skills and knowledge of parallel and distributed processing algorithms at even the earlier stages of an undergraduate career."

Consider your learning experience in the Introduction to Computer Systems class. Discuss the above claim! Do you agree or disagree? Justify your opinion!

Note: *This question will be graded both for content/correctness of your technical points (12 pts) and for your ability to communicate effectively in writing (8 pts). Your answer should be well-written, organized, and clear.* ***Your answer must be legible – points will be deducted for parts that cannot be read with normal effort.***

Technical content (12 points):

12 points discretionary if they make sound arguments about the technical issues involved.

3 points – clear statement of position on question
3 points – clear statements of evidence/arguments in support of position
3 points – clear statements of evidence/arguments that might be complicating/mitigating factors OR clear statements indicating the importance of the issues of parallel/distributed programming

Writing (8 points):

3 points for overall structure
3 points for expressing ideas clearly
2 points for correct grammar/spelling

6. Essay Question: The Problem With Threads (20 points)

Edward Lee criticizes the thread model because it leads to errors caused by unpredictable interleaving (non-determinism) of execution by different threads. Lee calls it “pruning” when developers use locks, semaphores or other techniques to control the non-determinism. Lee writes:

“Although threads seem to be a small step from sequential computation, in fact, they represent a huge step. They discard the most essential and appealing properties of sequential computation: understandability, predictability, and determinism. Threads, as a model of computation, are wildly nondeterministic, and the job of the programmer becomes one of pruning that nondeterminism. Although many research techniques improve the model by offering more effective pruning, I argue that this is approaching the problem backwards. Nondeterminism should be explicitly and judiciously introduced where needed, rather than removed where not needed. Threads must be relegated to the engine room of computing, to be suffered only by expert technology providers.”

Consider your learning experience in the Introduction to Computer Systems class. Discuss the above claim! Do you agree or disagree? Justify your opinion!

Note: *This question will be graded both for content/correctness of your technical points (12 pts) and for your ability to communicate effectively in writing (8 pts). Your answer should be well-written, organized, and clear.*

Technical content:

12 points discretionary if they make sound arguments about the technical issues involved.

Writing:

3 points for overall structure
3 points for expressing ideas clearly
2 points for correct grammar/spelling