This is a practice exam for CS 241. This exam comes with no guarantees regarding its similarity to the actual final exam. Questions were derived from material in past semesters, so for some topics, it may not reflect the same time or depth with which the topic was covered this semester. In addition, we have not vetted this practice final with the same rigor as the actual final, so there is a chance of mistakes.

TIME: 1:30pm – 4:30pm
DATE: Friday, May 11, 2012

INSTRUCTIONS

1. Put your full name and NetID on the bubble sheet and on the front of this booklet.
2. This is a closed book exam. You may not consult any materials during the exam other than the exam itself: no textbooks, no crib sheets, no calculators, etc
3. Cheating or apparent cheating in any form will be taken very seriously.
4. Cell phones and other electronic devices must be turned off and stored.
5. Time: 180 minutes
6. The exam proctors will announce when there are 10 minutes left.
7. When instructed to do so, stop writing, put your pencils down, and hand in your bubble answer sheet and exam to an exam proctor.
8. There are 24 questions multiple choice questions and eight (8) free-response questions.
9. All true/false and multiple choice questions only have one BEST ANSWER.
10. For true/false and multiple choice questions, answer the questions on the bubble sheet. Carefully shade answers in pencil.
11. For long answer questions, please write in the appropriate spaces in the exam booklet. Please write legibly. If the course staff cannot read your solution, no credit will be given.
12. PLEASE read the questions carefully. For example, some of the questions ask you to select the WRONG (INCORRECT) statement from a list of statements.
13. Unless otherwise stated, assume any machine-specific details are those of the EWS Linux boxes that are for your use on MPs (64-bit addresses where **`sizeof(int) == 4`**, **`sizeof(double) == 8`**, and **`sizeof(char) == 1`**).

When instructed to do so you may turn this page over, begin the exam and answer the questions.

Maximum Points: 143

	Max	**Score**	**Grader**
Part 1	**48**		
Part 2, #1	**10**		
Part 2, #2	**15**		
Part 2, #3	**15**		
Part 2, #4	**10**		
Part 2, #5	**6**		
Part 2, #6	**9**		
Part 2, #7	**15**		
Part 2, #8	**15**		
Total	**143**		

Name: _____ NetID: _____

This page intentionally left blank

Section I: True/False

Answer all questions on the scantron form. Fill in A for TRUE and fill in B for FALSE.
Each question is worth 2 points.

1. The port number of a connection will tell you whether it is running UDP or TCP.

2. Using the TCP/IP protocol over an unreliable network, the receiving application will not receive the same packet multiple times while the connection is alive.

3. FIFO page replacement replaces the page that was the most recently swapped in.

4. While a process is blocked on a semaphore's queue, it is engaged in busy waiting.

5. The allocated portions of memory using a buddy system are not the same size.

6. Thrashing will never be a problem if the system has 1 GB of real memory.

7. TCP is a better choice than UDP for a chat client.

8. A memory block in the Buddy System can find its buddy in constant time.

9. Locks are ALWAYS required when memory is shared between threads.

10. SVN allows multiple users to backup work on the same project.

Section II: Multiple Choice

Answer all questions on the scantron form. Each question has ONE best answer.
Each question is worth 2 points.

11) Consider the following code segment:

```
void *ptr = malloc(1024 * sizeof(char));
printf("%p", ptr);
```

When this program is run as two separate processes, you notice the following output:

```
Process #1: 0x49301240
Process #2: 0xac382ac0
```

Based on the output above, what can be determined about the address contained in `ptr`?

A. The address of Process #1 is located before the address of Process #2 in physical memory.
B. The address of Process #2 is located before the address of Process #1 in physical memory.
C. The address of Process #1 and Process #2 is located in the same physical memory.
D. None of the above.

12) Consider a simple system with three (3) memory frames and the following sequence of frame accesses:

```
1 2 3 4 2 3 4 1 2 1 1
```

If the FIFO page replacement algorithm was used, what is the final configuration of the memory frames?

A. 1, 2, 3
B. 4, 1, 2
C. 3, 1, 2
D. 4, 2, 1

13) Using the same access pattern, if the Most Recently Used (MRU) replacement algorithm was used, what will the final configuration of the memory frames be?

A. 3, 4, 1
B. 1, 4, 2
C. 3, 2, 1
D. 1, 2, 3

14) Thrashing…

A. occurs whenever a process is spending more time paging than executing.
B. happens when page faults occur very frequently and constantly.
C. decreases system performance substantially.
D. All of the above

15) Which term below can be defined as "a code region that uses a shared resource that must not be concurrently accessed by more than one thread"?
    A. Deadlock
    B. Critical section
    C. Race condition
    D. Mutual exclusion
    E. Semaphore

16) Which one of the following features is NOT directly addressed in the TCP protocol?
    A. Encryption of application data
    B. Retransmission of lost packets
    C. Duplicate packet detection
    D. Ordered delivery

17) What happens when a POSIX thread calls **`exit()`**?
    A. All threads in the same process terminate
    B. Nothing, multithreaded programs must use **`pthread_exit()`**
    C. All threads are terminated except those that are waiting for a signal
    D. Only the calling thread terminates
    E. All threads are terminated except those that are waiting for I/O

18) Which one of the following is a reason that a process transitions from the Blocked state to the Ready state?
    A. The process has run until the end of the time-slice quantum
    B. A non-reentrant interrupt routine has been disabled for this process
    C. The process received SIGSTOP or one of the stop signals
    D. A disk request from the process to transfer data from disk is now complete
    E. A prioritized system interrupt routine must now be executed

19) In MP8, when the web server sets up its first socket, it must make all of the following function calls EXCEPT:
    A. listen()
    B. bind()
    C. connect()
    D. socket()

20) Which of the following is a valid solution to the Dining Philosopher's Problem (guarantees that some philosopher will eat)?
    A. Number the chopsticks. Pick up the chopstick with the lower value first, then the higher value
    B. Pick up the left chopstick first, then the right chopstick
    C. Pick up the right chopstick first, then the left chopstick
    D. Pick up the left chopstick. If the right chopstick is not on the table, set the left chopstick down. Repeat.
    E. None of the above

21) Which of the following statements about POSIX threads is NOT true?
   A. `pthread_create()` must start execution in a new function
   B. `pthread_join()` will obtain the thread's exit code if `pthread_join` succeeds
   C. `pthread_detach()` will ensure that a thread's state is freed upon exit
   D. It is impossible to pass a string as an argument to a new thread
   E. POSIX threads are not the only type of threads that exist

22) When a program tries to access an invalid memory location the operating system normally terminates the program. Consider a running program that tries to dereference a NULL pointer but is not terminated. Which of the following best describes the situation?
   A. The program is running with root privileges and so it cannot be terminated
   B. The program blocks all signals except SIGSEGV
   C. The program blocks SIGSEGV
   D. NULL is not an invalid memory location
   E. The program is multithreaded, thus only the thread that caused the violation is terminated, not the whole program

23) Which of the following best describes the page-out process of a page in memory?
   A. Write dirty pages to disk and reset the dirty bit.
   B. Write read-only pages to disk.
   C. Return unused pages to the memory pool.
   D. Read pages from disk as they are needed.
   E. Read pages from disk before they are needed.

24) Which one of the following describes one disadvantage of Shortest Job First scheduling?
   A. Suffers from the convoy effect.
   B. Suffers from deadlock for batch jobs.
   C. Always suffers from large average waiting times.
   D. Can result in starvation.
   E. Requires multi-level priority queues.

Section III: Long Answers

Problem 1 ***(10 points)***

Consider the following C function to create a new reversed copy of a string:

```
char *reverse(char *string)
{
   char *reversed;
   int idx;
   int length;

   if (!string) { return NULL; }

   /* Find length of the input. */
   (Line 1):____________________


   /* Allocate space for reversed string */
   (Line 2):____________________
   (Line 3):____________________

   /* Copy input to reversed string backwards */
   (Line 4):____________________
   (Line 5):____________________
   (Line 6):____________________


   /* Return the reversed string. */
   return reversed;
}
```

From the following lines of code, choose the line of code that best completes the functionality of reversing a string in the code above. Note that it may be necessary to use one line multiple times. Choose option **W** if no code is needed. Please enter the letter corresponding to your answers on the lines provided above.

```
A. string[idx] = reversed[idx];
B. reversed[idx] = string[idx];
C. reversed[length-idx - 1] =
              string[idx – 1];
D. reversed[idx] = string[length-idx];
E. strcpy(reversed, string);
F. reversed = strdup(string);
G. for(idx=0; idx < length; idx++)
H. for(idx=length; idx > 1 ; idx--)
I. for(idx=0; idx < length/2; idx++)
J. for(idx=0; idx < length; idx++,
              length--)
K. while(--length)
L. reversed[length] = '\0';
M. reversed = malloc(sizeof(char *));
N. reversed = malloc(length *
                   sizeof(char *));
O. reversed = malloc(sizeof(string) *
                   sizeof(char *));
P. reversed = malloc(strlen(string) *
                   sizeof(char));
Q. reversed = malloc(sizeof(char));
R. reversed = malloc(length *
                   sizeof(char));
S. reversed = malloc(length + 1 *
                   sizeof(char));
T. length = sizeof(string);
U. length = strlen(string);
V. length = strlen(string) + 1;
W. /* No code needed on this line. */
```

Problem 2 ***(15 points)***

Consider an operating system using virtual memory. Write the following function to decide if a given page is executable or not. The **EXEC_PERMISSION_BIT** is stored in the page table entry given as an argument (**`entry`**). If the bit is set (== 1) the page is executable, else the page is not executable. The function should return **`True`** if the page is executable, and **`False`** otherwise.

For example, if the page entry value is **`0x0000`**, the page is not executable. In 8 lines of code or fewer, write **compilable C code** that could be used in the **`check_executable()`** function to successfully check a page's permission before execution.

```
typedef enum{False=0, True=1) Bool;

enum{ EXEC_PERMISSION_BIT_MASK = 0x00000002};

Bool check_executable(long int entry) {









}
```

Hint: It is not necessary to use any library, function, or system calls.

Problem 3 ***(15 points)***
Suppose that we have the following resources: R1, R2, and R3 and threads T1, T2, and, T3. The maximum number of each resource available is:

Total Resources Available		
R1	**R2**	**R3**
8	5	3

Further, assume that the processes have the following maximum requirements and current allocations:

Thread ID		Current Allocation				Maximum Requirements		
		R1	**R2**	**R3**		**R1**	**R2**	**R3**
T1		2	1	0		4	1	1
T2		2	0	1		3	4	3
T3		1	1	1		5	2	1

The above state is safe. Consider each of the following requests and say if they can be granted. Assume each request starts from the above safe state. To receive full credit, show your work in the space provided by applying the Banker's algorithm. (*It may not be necessary to fill in every step to determine your answer.)*

1. (*5 points*) T3 makes a request for additional resources: (R1, R2, R3) = (2, 0, 0).

Step #1

Step #2

Step #3

Step #4

Can the request be granted?

[This question continues on the next page!]

2. (*5 points*) T2 makes a request for additional resources: (R1, R2, R3) = (0, 2, 0).

Step #1

Step #2

Step #3

Step #4

Can the request be granted?

3. (*5 points*) T1 makes a request for additional resources: (R1, R2, R3) = (0, 0, 1).

Step #1

Step #2

Step #3

Step #4

Can the request be granted?

Problem 4 ***(10 x 1 = 10 points)***

In an i-node based file system, pointers to disk blocks containing data are stored in the very last portion of an i-node. For this question, consider an i-node based file system with 4KB blocks and 4B disk addresses. Each i-node in this file system contains exactly 5 direct entries, 2 single indirect entry, 2 double indirect entry, and 1 triple indirect entry. Assume that the i-node itself takes an entire disk block.

In lecture, you were made familiar with the concepts of direct and indirect entries. The specific forms of the indirect entries in the i-node are detailed below, in the exact same way they are presented in the course.

A single indirect entry is a disk block that contains only a list of pointers to disk blocks which contain data. If a disk block is able to store 100 disk pointers, a single indirect entry would be able to store a file up to a size of 100 times that of a direct entry.

Likewise, a double indirect entry is a disk block that contains only a list of pointers to disk blocks which contain single indirect entries.

Finally, a triple indirect entry is a disk block that contains only a list of pointers to disk blocks which contain double indirect entries. Since each level of indirection requires more lookups to get to the raw data blocks, direct entries will always be filled before single indirect, single indirect entries will always be filled before double indirect, and triple indirect entries will always be filled last.

Problem 3, Part A

What is the maximum size file (be sure to include units) that can be stored using:

(i) only direct entries?

_____

(ii) only the single indirect entry? (do NOT add the direct entries to this answer)

_____

(iii) only the double indirect entry? (do NOT add single indirect or direct entries to this answer)

_____

(iv) only the triple indirect entry? (do NOT add double indirect, single indirect or direct entries to this answer)

_____

Hint: Each answer should be significantly larger than the previous answer.

[This question continues on the next page!]

Problem 4, Part B

Consider storing a 10 MB file using the file system described above.
(It may be useful to provide the answer in the form of 24 MB + 34 KB. An expression such as this will be considered CORRECT if the expression, when calculated, yields the correct answer.)

(i) How many direct entries are used in referencing this file?

_____

(ii) How much of the 10 MB file will be referenced by only direct entries?

_____

(iii) How much of the 10 MB file will be referenced through only the single-indirect entry?

_____

(iv) How much of the 10 MB file will be referenced through only the double-indirect entry?

_____

(v) How much of the 10 MB file will be referenced through only the triple-indirect entry?

_____

(vi) What is the total amount of overhead (in disk blocks) used by this i-node based file system? Include only the i-node itself and the indirect blocks used. The indirect blocks may include the single indirect, double indirect (and its single indirects), and triple indirect (and its double and single indirects). All other blocks that are not included in this answer would be storing the data of the 10 MB file.

_____ disk blocks

Problem 5 ***(6 points)***

A process is allocated 4 frames, initially empty. For each of the following page replacement algorithms, determine which pages are in memory after the last page is referenced and give the total number of page faults. The tables are given for you to work out your solutions. However, we will only grade the answers in the final column and your answer for the total number of page faults.

(i) **[2 points]** FIFO

Referenced Page	**1**	**2**	**1**	**2**	**40**	**39**	**38**	**37**	**2**	**4**	**5**	**38**	**1**
Frame 1													
Frame 2													
Frame 3													
Frame 4													
Fault?													

Total number of page faults = _____

(ii) **[2 points]** Optimal – fewest possible page faults (break ties with lower page number)

Referenced Page	**1**	**2**	**1**	**2**	**40**	**39**	**38**	**37**	**2**	**4**	**5**	**38**	**1**
Frame 1													
Frame 2													
Frame 3													
Frame 4													
Fault?													

Total number of page faults = _____

(iii) **[2 points]** LRU

Referenced Page	**1**	**2**	**1**	**2**	**40**	**39**	**38**	**37**	**2**	**4**	**5**	**38**	**1**
Frame 1													
Frame 2													
Frame 3													
Frame 4													
Fault?													

Total number of page faults = _____

Problem 6 ***(9 points)***

Consider a memory allocator, similar to the **`malloc()`** program you wrote. Instead of increasing the size of the heap, your system has a fixed heap size of 14 MB. Consider the following code:

```
void *p1 = malloc(2MB);
void *p2 = malloc(6MB);
void *p3 = malloc(1MB);
free(p2);
void *p4 = malloc(5MB);
free(p1);
void *p5 = malloc(2MB);
```

Show the heap allocation after the above calls, using a best-fit, worst-fit, and first-fit algorithm. You should:

- Shade the cells of the memory regions marked as used by your allocator,
- Identify the value pointed to by **`p4`** and **`p5`** by writing their locations in the blanks provided, **AND**
- Determine the largest available contiguous free space available at the end of the allocation.

You may assume that all memory is initially freed and no metadata is required to maintain any sort of data structure. Memory should only be allocated in 1 MB chunks, on the 1 MB boundaries (eg: no half filled in boxes). *Note: The cell marked "0 MB" corresponds to the memory segment [0 MB, 1 MB).*

1. (3 *points*) Best-Fit:

(MB)	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Working Space (Not Graded)														
Final State: (Graded)														

`p4`: ____ MB **`p5`**: ____ MB Largest contiguous free space: _____ MB

2. (*3 points*) Worst-Fit:

(MB)	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Working Space (Not Graded)														
Final State: (Graded)														

`p4`: ____ MB **`p5`**: ____ MB Largest contiguous free space: _____ MB

3. (3 *points*) First-Fit:

(MB)	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Working Space (Not Graded)														
Final State: (Graded)														

`p4`: ____ MB **`p5`**: ____ MB Largest contiguous free space: _____ MB

Problem 7 ***(15 points)***

Consider a web browser on host X with a cached IP address for a target URL (i.e. you can ignore DNS lookups). The Round-Trip Time (RTT) is the time to send a packet from X to the server and back. The server will send a single HTTP Response after which it will close the connection. Assume the transmission times for all of the objects are equal and included in the RTT. How much time elapses (in terms of RTTs) from when the user enters the URL until the connection is closed? Include an itemized list.

1. ***(15 points)***

Problem 8 ***(15 points)***

You have been asked to use semaphores to improve the efficiency of a web browser. Design an efficient and elegant semaphore solution. You have to complete two procedures, `makeHttpRequest()` and `processAllHttpReqeuests()`, which are used as described below.

- M requester threads: each thread repeatedly calls the following procedure: `makeHttpRequest(requesterId, url)`: the `makeHttpRequest()` procedure should queue urls in the shared buffer and then block until the delivery request has been serviced by the service thread.

- One service thread: the thread repeatedly calls the following procedure: `processAllHttpRequests()`: the `processAllHttpRequests()` procedure should block until there are at least N deliver requests in the shared buffer. Then it should service all the queued requests (N or more), emptying the queue.

- shared buffer between requesters and the service thread: unlimited in size.

Clearly indicate all the shared variables, locks, and semaphores that you use in your solution, as well as any initial values. Points will be awarded for correctness, simplicity of the solution, and efficiency (avoid busy waiting). Pseudo-code is fine as long as the usage of semaphore primitives is clear and the algorithm is clear.

[All code should go on the next page!]

Problem 8, Coding

```
/* global shared data goes here */
queue urlData;  // supports size(), add(), remove(), and any other queue
                // operations you need




/* global locks and semaphores go here */




makeHttpRequest(requesterID, url)
{




   urlData.add(requesterID, url);




}

processAllHttpsRequeusts() {



   /* service all the requests in the queue */
   while (urlData.size() != 0) {



      ... <requesterID, url> = urlData.remove(); ...
      ... service request from requesterID for url; ...


   }



}
```

This page intentionally left blank

This page intentionally left blank