INSTRUCTIONS

This practice exam is a close replica to the Fall 2011 Midterm Exam. Some questions have been added or removed to match this semester's material closer. You can expect your exam to have a similar form although the exact questions will be different. It is suggested that you take time to review topics BEFORE taking this exam. Treat this practice test as if it were the actual exam by eliminating distractions and timing yourself.

Remember:

- **Bring a pencil and eraser**
- **No cheat sheets, calculators, or any other form of data storage besides your brain**

TIME: 7:00 pm – 9:00 pm

DATE: Tuesday March 6, 2012

INSTRUCTIONS

1. Put your last 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: 120 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 33 questions – the first 5 are true/false questions worth 1 point each, the next 14 questions are multiple choice questions worth 2 points each, the next 9 questions are multiple choice questions worth 3 points each, and the last 5 questions require written answers in the exam booklet and are worth 8 points each.
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: 100

	Max	Score	Grader
Scantron	**60**		
Q. 29	**8**		
Q. 30	**8**		
Q. 31	**8**		
Q. 32	**8**		
Q. 33	**8**		
Total	**100**		

Name: _____ NetID: _____

This page intentionally left blank

Part I – 5 TRUE/FALSE questions – 1 point per question: Total 5 points

1. Round Robin scheduling guarantees that all threads in the scheduler queue will eventually be scheduled?

    A. True
    B. False

2. The scheduler forced a process to switch from running to ready state. The scheduling algorithm must be non-preemptive:

    A. True
    B. False

3. The function **`pthread_create()`** is similar to **`fork()`** in that the new thread starts from the code immediately following **`pthread_create()`**.

    A. True
    B. False

Part II – 14 MULTIPLE CHOICE questions – 2 points per question: Total 28 points

4. Which one of the following correctly describes one difference between system calls to the kernel and calls to a user library?
   A. Only system calls can write to user memory.
   B. User library calls do not maintain stack pointers.
   C. Only user library calls can be called from a signal handler.
   D. User library calls cannot be called from a background thread.
   E. System calls are invoked using an interrupt.

5. Consider the following code snippet:

```
myStruct myArray[10];

myStruct *ptr = myArray;
```

Which one of the following will always point to **`myArray[1]`**?

A. **`ptr = ptr + 1`**
B. **`ptr = ptr + sizeof(myStruct*)`**
C. **`ptr = ptr + sizeof(myArray)`**
D. **`ptr = ptr + sizeof(ptr)`**
E. None of the above

6. Which of the following scheduling algorithms will have the LONGEST average response time after many jobs are queued and ran to completion?

   A. Round Robin with a quantum of much less than the shortest job
   B. Shortest Job First (SJF)
   C. Preemptive Shortest Job First (PSJF)
   D. First Come First Serve (FCFS)

7. The C code printed below is an incomplete version of the C string library call **`strlen()`**.

```
unsigned int strlen(const char *s)
{
  /* Your code here. */
}
```

Which line of code creates a working and accurate version of **`strlen()`**?

A. **`return sizeof(s);`**
B. **`return sizeof(s) - 1;`**
C. **`int i = 0;  while (s[i++] != NULL);  return i;`**
D. **`int i = 0;  while (s[i++] != NULL);  return i - 1;`**
E. **`int i = 0;  while (i++ != sizeof(s));  return i;`**

8. Consider the following code snippet. What is the output?

```
int sum(int a) {
    static int total = 0;
    total += a;
    return total;
}
int main() {
    int a;
    a = sum(1);
    a += sum(2);
    a += sum(3);
    printf("sum is %d\n", a);
    return 0;
}
```

A. **sum** is 6
B. **sum** is 7
C. **sum** is 9
D. **sum** is 10
E. None of the above

9. Which one of the following scheduling algorithms may NOT lead to starvation?

A. First Come First Serve (FCFS)
B. Shortest Job First (SJF)
C. Priority (PRI)
D. Preemptive Shortest Job First (PSJF)
E. All of the above lead to starvation

10. The sample code below releases memory from a linked list. Which of the choices below accurately describes how it will work?

```
/* pointers for a linked list. */
struct node *currPtr, *startPtr = create_list();

for (currPtr=startPtr; currPtr != NULL;
     currPtr=currPtr->next)
     free(currPtr);
```

A. It will work correctly since the for loop covers the entire list
B. It may fail since each node **currPtr** is freed before its next address can be accessed
C. In the for loop, the assignment **currPtr=currPtr->next** should be changed to **currPtr=currPtr.next**
D. The loop will never end

11. Which one of the following is true for Shortest Job First scheduling?

    A. Minimizes average waiting time and minimizes turnaround time.
    B. Exhibits the convoy effect.
    C. Minimizes CPU throughput.
    D. Can be approximated with Round Robin scheduling with a long time slice quantum.

12. What are the advantages of implementing threads in the kernel?

    A. No run-time system is needed in each process
    B. Fewer kernel resources are used
    C. Threads are only available in non-preemptive Operating Systems.
    D. A thread can make blocking system calls without affecting other threads of the same process
    E. Fewer CPU cache lines are required

13. Which of the following pieces of information IS SHARED among all threads in a single process?

    A. The heap memory.
    B. The stack memory.
    C. The stack memory in the **`main()`** function only.
    D. The program counter.
    E. None of the above information IS SHARED among all threads.

14. Consider the following code:

```
void *hello_thread(void *arg) {
   printf("Hello\n");  return NULL;
}

void *world_thread(void *arg) {
   printf("World\n"); return NULL;
}

int main(int argc, char **argv) {
   pthread_t hello, world;
   pthread_create(&hello, NULL, hello_thread, NULL);
   pthread_create(&world, NULL, world_thread, NULL);
   pthread_join(world, NULL);
   return 0;
}
```

What would be the output of the program:

A. only "**World\n**"
B. "**Hello\nWorld\n**" or "**World\nHello\n**"
C. "**World\n**" or "**Hello\nWorld\n**" or "**World\nHello\n**"
D. "**Hello\n**" or "**World\n**"
E. "**Hello\n**" or "**Hello\nWorld\n**" or "**World\nHello\n**"

15. Which of the following causes a POSIX **process** (not just the calling thread) to **exit**:

A. A call to **`pthread_exit`** by a child thread but not the main thread.
B. A call to **`pthread_exit`** by the main thread.
C. A call to **`pthread_join`** by the main thread that returns after a child thread (not necessarily the last one) has exited.
D. A call to **`pthread_join`** by the main thread that returns after the last child thread has exited.
E. None of the above.

16. After a large number of requests, which memory allocation algorithm is likely to result in the least variance between memory hole sizes (eg: most memory holes are about the same size)?

A. First Fit
B. Next Fit
C. Best Fit
D. Worst Fit
E. Random Fit

Part III – 9 questions - 3 points per question: Total 27 points

17. Consider the workload in the following table:

Process	**Execution Time**	**Priority***	**Arrival Time**
P1	8ms	2	0ms
P2	1ms	3	1ms
P3	4ms	4	2ms
P4	4ms	1	4ms
		**: Lower number means higher priority*	

The following schedule is an example of which scheduling policy?

Time: (ms)	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Job:	P1	P1	P1	P1	P4	P4	P4	P4	P1	P1	P1	P1	P2	P3	P3	P3	P3

A. First-Come-First-Served
B. Shortest-Job-First
C. Preemptive Priority-based scheduler
D. Non-Preemptive Priority-based scheduler
E. Round-Robin (quantum = 6ms)

18. 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.

This page intentionally left blank

Part III – 5 questions - 8 points per question: Total 40 points

19. Implement in the following space the `init()`, `insert()`, and `lookup()` functions for a `vector`. You may assume that `init()` is called only once before any calls to `insert()` or `lookup()`. New elements should be inserted at the end of the vector and the vector starts at 0. (Hint: This can be done as a linked-list). You do not need to add any error checking code.

```
typedef struct _vector_t{

} vector_t;

typedef struct _node_t{

} node_t;

void vector_init (vector_t * v) {



}

void vector_insert (vector_t *  v, void * element) {









}

void *vector_lookup (vector_t* v, int position) {









}
```

20. Given the following job arrival time, length and priority, finish the following questions. Smaller priority value means higher priority. If there is a tie, schedule process with lower pid first. Jobs arrive in the following order:

	pid	Arrival Time	Duration	Priority
Process A	241	0	3	5
Process B	242	0	5	4
Process C	243	3	2	3
Process D	244	4	2	0

A. (6 points) Fill in start time and finish time for each process in the table below (PPRI = Preemptive Priority Scheduler, PSJF = Preemptive Shortest Job First). The system starts at time 0 and ends at time 12.

	PPRI Start Time	PPRI Finish Time	PSJF Start Time	PSJF Finish Time
Process A				
Process B				
Process C				
Process D				

B. (2 points) Calculate the average wait time for each scheduler. Show your work! You can leave your answers in fractional form.

i. Average wait time for Preemptive Priority Scheduler:

ii. Average wait time for Preemptive Shortest Job First:

21. The code printed below is shared between multiple threads and is run concurrently. Using any techniques learned in CS 241, modify the code to ensure that it will always run properly. Provide any comment regarding initialization of variables.

```
int count = 0;

int increment (void) {

     count++;

     if (count > 5) {

          printf("counter %d reached value > 5", count);

          return 0;

     }

     return 1;

}

int decrement (void) {

     while (count > 5) {

          printf("counter %d is > 5:", count);

          count --;

     }

     if (count == 0) {

          return 0;

     }

     return 1;

}
```

22. Consider the following C function to duplicate the contents of a string.

```
/*    Returns a pointer to a new string which is a duplicate of the
string s.   */

char *strdup (const char *s)   {

    // create a new string
      (Line 1):_____________________________________________

    // copy string content
      (Line 2):_____________________________________________

    // return string
      (Line 3):_____________________________________________

}
```

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

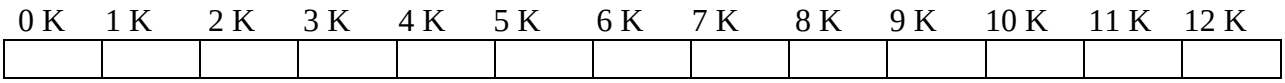
```
A. char *ptr = (char *) malloc(strlen(s));
B. char ptr[strlen(s) + 1];
C. char *ptr = (char *) malloc((sizeof(s) + 1)
                               * sizeof(char *));
D. char *ptr = (char *) malloc(strlen(s) + 1);
E. strncpy(ptr, s, strlen(s));
F. strcpy(ptr, s); ptr[strlen(s) - 1] = 0;
G. for (i = 0; i < sizeof(s); i++)  ptr[i] = s[i];
H. for (i = 0; i < strlen(s) + 1; i++)  ptr[i] = s[i];
I. return ptr;
J. return s;
K. return &s;
L. return *ptr;
M. /* no code needed on this line. */
```

33. Suppose the heap is managed with a linked list. Each node in the list is either allocated or free. The list is sorted by address. When **malloc()** is called, the list is searched for a free segment that is big enough (depending on the allocation algorithm), that segment is split into an allocated segment (at the beginning) and a free segment. When **free()** is called, the corresponding segment should merge with its neighboring segments, if they are also free. Now suppose a process has a heap of 13KB, which is initially unallocated. During its execution, the process issues the following memory allocate/de-allocate calls (**p1... p5** are **void*** pointers). In all cases, break ties by choosing the earliest segment. Also, assume all algorithms allocate memory from the beginning of the free segment they choose.

p1 = malloc(3KB)
p2 = malloc(4KB)
p3 = malloc(3KB)
free(p2)
p4 = malloc(3KB)
free(p1)
p5 = malloc(1KB)

For simplicity, assume the memory begins at address 0, and ignore the memory used by the linked list itself. Show the heap allocation after the above calls, using best-fit, worst-fit and first-fit algorithms respectively. To help you work out the problems, you can shade in which 1K blocks of memory are allocated for each algorithm. However, these diagrams will not be graded.

Best Fit:**[2 points]**



- Identify the starting address of p4 and p5.

p4 = _____

p5 = _____

Worst Fit:**[2 points]**

0 K	1 K	2 K	3 K	4 K	5 K	6 K	7 K	8 K	9 K	10 K	11 K	12 K

- Identify the starting address of p4 and p5.

p4 = _____

p5 = _____

First Fit:**[2 points]**

0 K	1 K	2 K	3 K	4 K	5 K	6 K	7 K	8 K	9 K	10 K	11 K	12 K

- Identify the starting address of p4 and p5.

p4 = _____K

p5 = _____K

This page intentionally left blank