CS-450 & CS-550: Operating Systems

Fall 2008: Mid-Term Exam *ANSWERS*

CORRECTED

Rules for the Examination:

1. The JMU Honor Code applies. You will not get credit for your grade unless you sign the Honor Code declaration (the Honor Code declaration is a JMU requirement). Please sign below ***legibly*** to indicate your compliance with the Honor Code:

   16 Oct 2008 Signature: _____

2. You must also write your name legibly ***on the back of the last page*** of this examination, and nowhere else, so that I know who you are. Please also write the **last four digits only** of your JMU ID number in the indicated location at the top of each odd-numbered page.
3. This examination is **closed book, closed notes** (but open mind), ***NO*** **calculators** allowed**.**
4. For many of the questions, you are given several choices for the answer, with each choice bearing an identifying letter. In some questions, you may be asked to write in the answer. To get credit for your work, you **must** mark the answer to each question **on the answer sheet that is provided at the end of the examination,** by indicating the ***lettered choice*** next to the number for the question, where such choices are provided to you.
5. Where lettered choices are not enumerated in the question, fill in the answer in the space provided.
6. All work is to be done on the examination pages, and is to be handed in along with the answer sheet. You will not receive credit for any answer that needs to be calculated or worked out, unless the justification for your answer is clearly apparent from your work.

Example: Indicate the name of the capital city for each country:

Question #:	Country:
101	England
102	France
103	Germany
104	Greece
105	Turkey

Choice:	Capital City
A	Istanbul
B	London
C	Berlin
D	Paris
E	Athens
F	none of the above

Answer: On your answer sheet **at the end of the examination,** you would mark the correct answers as follows:

Question #:	Answer:
101	**B**
102	**D**
103	**C**
104	**E**
105	**F**

Item #i: The following statements all pertain to the Kernel of the Operating System. They are general statements, and you should consider their truth or falsity on the basis of whether or not they apply in general to all or nearly all Operating Systems, not just to one in particular. Mark each statement either **T** (true) or **F** (false):

2 pts each

1. The kernel implements all of the functionality of the Operating System. *Answer*: **F**alse
2. All of the kernel of the Operating System must be continuously memory-resident. *Answer*: **T**rue
3. The kernel of the Operating System includes the Interrupt Vector. *Answer*: **T**rue
4. External Fragmentation and Internal Fragmentation of main memory can both be present simultaneously in a single computer system. *Answer*: **F**alse
5. On a multi-user computer system, the Operating System is responsible for preventing a user from accessing the contents of system memory and of secondary storage allocated to other users. *Answer*: **T**rue
6. In multi-threaded processing with Kernel-Level threads, the various threads associated with a single process must share a common Thread State (*e.g.*, Ready, Blocked, or Running). *Answer*: **F**alse
7. Part of the kernel is memory-resident, and part of it is not. *Answer*: **F**alse
8. In all operating systems, a new program can be loaded into a child process's Memory Address space **at the instant of initial creation of the child process,** different from the program in the parent's Memory Address space, if it is desired that the child process execute a different program than the parent. *Answer*: **F**alse
9. For Kernel-Level threads, when one thread associated with a particular process becomes blocked, then the entire process is blocked. *Answer*: **F**alse
10. The kernel is the part of the Operating System that runs continuously. *Answer*: **F**alse
11. A mid-term scheduler must be part of every Operating System other than those that execute a single-task-at-a-time. *Answer*: **F**alse
12. In multi-threaded processing, the various threads associated with a single process share a common User Stack. *Answer*: **F**alse

Item # ii: Indicate which of the following items are organized as separate processes (**Y**es or **N**o):

2 pts each

13. Several print jobs in the print SPOOLer. *Answer*: **N**o
14. The service routine for an interrupt. *Answer*: **N**o
15. Calls from within a single application program to several different subroutines. *Answer*: **N**o
16. Each different application program running on the system. *Answer*: **Y**es

Item # iii: Indicate which of the following are System Administration tasks (**Y**es or **N**o):

2 pts each

17. Addition of a new user. *Answer*: **Y**es
18. Deletion of a user. *Answer*: **Y**es
19. Installation of new application software. *Answer*: **Y**es
20. Upgrading of existing application software. *Answer*: **Y**es
21. Performance of secure backup. *Answer*: **Y**es

Item # iv: Which of the following are included in a process image (**Y**es or **N**o):

2 pts each

22. Executable program code. *Answer*: **Y**es
23. The values of variables. *Answer*: **Y**es
24. Copies of the contents of the various registers as of the last time the process ran on the CPU. *Answer*: **N**o
25. The user stack. *Answer*: **Y**es
26. The Process Control Block (PCB). *Answer*: **N**o

Item # v: Consider the following **direct** process state transitions. By **direct** it is meant that a process can transition directly from the first state to the second without passing through any intermediate state. Indicate which of the direct state transitions listed can or cannot occur (**Y**es or **N**o):

2 pts each

27. Running → Ready *Answer*: **Y**es
28. Ready → Running *Answer*: **Y**es
29. Ready → Blocked *Answer*: **N**o
30. Blocked → Ready *Answer*: **Y**es
31. Running → Blocked *Answer*: **Y**es
32. Blocked → Running *Answer*: **N**o

Item # vi: With regard to scheduling, indicate into which category of scheduling, if any, each of the activities listed below falls:

2 pts each

33. Dispatching of a Kernel-Level Thread to the CPU *Answer*: **C**
34. Dispatching of a User-Level Thread to the CPU. *Answer*: **C**
35. Dispatching of a process to the CPU. *Answer*: **C**
36. Swapping of a process in to main memory from the disk or out of main memory to the disk. *Answer*: **B**
37. Admission of a process to the system. *Answer*: **A**

A. Long-Term Scheduling
B. Mid-Term Scheduling
C. Short-Term Scheduling
D. all of the above.
E. more than one, but **not** all, of the above.
F. none of the above.

Item vii: Indicate which statements are **T**rue, and which are **F**alse, regarding a single process and its threads:

2 pts each

38. In an operating system environment that operates with User-Level threads only (no Kernel-Level threads), the various threads associated with a single process all share a common context, including a common PC. *Answer*: **F**alse
39. In an operating system environment that operates with Kernel-Level threads only (no User-Level threads), the various threads associated with a single process all share a common context, including a common PC. *Answer*: **F**alse
40. The different threads associated with a single process all share a single process**or** context., *i.e.*, the value of the **P**rocessor **S**tatus **W**ord is the same for all such threads. *Answer*: **F**alse
41. Thread creation takes more time than process creation. *Answer*: **F**alse
42. Each thread associated with a single process has exclusive ownership of any files that it has opened; they are not available to other threads associated with the same process. *Answer*: **F**alse
43. Each thread associated with a single process has a separate text (program code) area. *Answer*: **F**alse

Item viii: A **Preemptive** job scheduler is invoked when (**Y**es or **N**o):

2 pts each

44. the currently running process finishes execution and terminates. *Answer*: **Y**es
45. the currently running process issues a blocking system call. *Answer*: **Y**es
46. another process's I/O completes, moving that process to the ready queue. *Answer*: **Y**es
47. a Timer Interrupt occurs: time quantum exceeded. *Answer*: **Y**es

Item ix: Consider the following page reference stream:

0, 3, 1, 4, 1,5, 1, 6, 0, 5, 2, 6, 7, 5, 0, 0, 0, 6, 6, 6, 6, 3, 2, 4, 3, 4

For a page frame allocation of **4**, and assuming that the primary memory is initially unloaded, determine how many page faults will this page reference stream incur under:

12 pts each

48. . Belady's Optimal Algorithm.
49. the First-In-First-Out (FIFO) algorithm.
50. the Least Recently Used (LRU) algorithm.

OPTIMAL ALGORITHM																											
Page Referenced	**0**	**3**	**1**	**4**	**1**	**5**	**1**	**6**	**0**	**5**	**2**	**6**	**7**	**5**	**0**	**0**	**0**	**6**	**6**	**6**	**6**	**3**	**2**	**4**	**3**	**4**
Page Fault	**F**	**F**	**F**	**F**	**N**	**F**	**N**	**F**	**N**	**N**	**F**	**N**	**F**	**N**	**N**	**N**	**N**	**N**	**N**	**N**	**N**	**F**	**F**	**F**	**N**	**N**
Page Frame 1	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**3**	3	3	3	3
Page Frame 2		**3**	3	3	3	3	3	3	3	3	**2**	2	**7**	7	7	7	7	7	7	7	7	7	7	7	7	7
Page Frame 3			**1**	1	1	1	1	**6**	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	**4**	4	4
Page Frame 4				**4**	4	**5**	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	**2**	2	2	2

Answer: **11** page faults (4 + 7)

First-In-First-Out (FIFO) ALGORITHM																											
Page Referenced	**0**	**3**	**1**	**4**	**1**	**5**	**1**	**6**	**0**	**5**	**2**	**6**	**7**	**5**	**0**	**0**	**0**	**6**	**6**	**6**	**6**	**3**	**2**	**4**	**3**	**4**
Page Fault	**F**	**F**	**F**	**F**	**N**	**F**	**N**	**F**	**F**	**N**	**F**	**N**	**F**	**F**	**N**	**N**	**N**	**F**	**N**	**N**	**N**	**F**	**F**	**F**	**N**	**N**
Page Frame 1	**0**	0	0	0	0	**5**	5	5	5	5	5	5	**7**	7	7	7	7	7	7	7	7	7	**2**	2	2	2
Page Frame 2		**3**	3	3	3	3	3	**6**	6	6	6	6	6	**5**	5	5	5	5	5	5	5	5	5	**4**	4	4
Page Frame 3			**1**	1	1	1	1	1	**0**	0	0	0	0	0	0	0	0	**6**	6	6	6	6	6	6	6	6
Page Frame 4				**4**	4	4	4	4	4	4	**2**	2	2	2	2	2	2	2	2	2	2	**3**	3	3	3	3

Answer: **14** page faults (4 + 10)

Least-Recently-Used (LRU) ALGORITHM																										
Page Referenced	**0**	**3**	**1**	**4**	**1**	**5**	**1**	**6**	**0**	**5**	**2**	**6**	**7**	**5**	**0**	**0**	**0**	**6**	**6**	**6**	**6**	**3**	**2**	**4**	**3**	**4**
Page Fault	**F**	**F**	**F**	**F**	**N**	**F**	**N**	**F**	**F**	**N**	**F**	**N**	**F**	**N**	**F**	**N**	**N**	**N**	**N**	**N**	**N**	**F**	**F**	**F**	**N**	**N**
Page Frame 1	**0**	0	0	0	0	**5**	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	**2**	2	2	2
Page Frame 2		**3**	3	3	3	3	3	**6**	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
Page Frame 3			**1**	1	1	1	1	1	1	1	**2**	2	2	2	**0**	0	0	0	0	0	0	0	0	**4**	4	4
Page Frame 4				**4**	4	4	4	4	**0**	0	0	0	**7**	7	7	7	7	7	7	7	7	**3**	3	3	3	3

Answer: **13** page faults (4 + 9)

Item x: Several processes enter the ready queue in the order shown. Each I/O takes 20 msec, and I/Os are executed on a separate, dedicated device for each process, so that the I/Os for different processes can run simultaneously. Context switching takes less than 100 μsec, and can therefore be ignored. **Lower** priority numbers correspond to **higher** process priority. Assume that the CPU Burst Time for each process is identical from one burst to the next:

Time of Arrival	**Process ID**	**CPU Burst Time (msec)**	**Process Priority (useful <u>only</u> with regard to Priority Scheduling)**
0.0 msec	P1	14	3
1.0 msec	P2	7	7
2.0 msec	P3	27	2
3.0 msec	P4	4	3
4.0 msec	P5	10	1

For each processor-scheduling algorithm indicated, determine **which** process does the scheduler assign to the processor for the sixth **scheduling decision**, and also at what time is that scheduling decision made.

<u>8</u> *pts each*

Algorithm	**Process**	**Start Time**
Shortest Remaining Time Next	**51.**	**52.**
Round Robin (time quantum = 12 msec)	**53.**	**54.**
Priority Scheduling **without** preemption (simple priority: NO aging, NO adjustments of any kind in process priority)	**55.**	**56.**

Scheduling Algorithm (qualifier): **Shortest Remaining Time Next**

Time: 0 msec. Process P1 runs.						
READY Queue				**WAIT Queue**		
Element	**PID**	**RBT**		**Element**	**trrq**	**PID**
First:	**P1**	**14**		First:		
Second:				Second:		
Third:				Third:		
Fourth:				Fourth:		
Fifth:				Fifth:		

Time: 1 msec. Process P2 runs.						
READY Queue				**WAIT Queue**		
Element	**PID**	**RBT**		**Element**	**trrq**	**PID**
First:	**P2**	**7**		First:		
Second:	**P1**	**13**		Second:		
Third:				Third:		
Fourth:				Fourth:		
Fifth:				Fifth:		

Time: 2 msec. Process P2 runs.						
READY Queue				**WAIT Queue**		
Element	**PID**	**RBT**		**Element**	**trrq**	**PID**
First:	**P1**	**13**		First:		
Second:	**P3**	**27**		Second:		
Third:	**P2**	**6**		Third:		
Fourth:				Fourth:		
Fifth:				Fifth:		

Time: 3 msec. Process P4 runs.						
READY Queue				**WAIT Queue**		
Element	**PID**	**RBT**		**Element**	**trrq**	**PID**
First:	**P1**	**13**		First:		
Second:	**P3**	**27**		Second:		
Third:	**P4**	**4**		Third:		
Fourth:	**P2**	**5**		Fourth:		
Fifth:				Fifth:		

Time: 4 msec. Process P4 runs.						
READY Queue				**WAIT Queue**		
Element	**PID**	**RBT**		**Element**	**trrq**	**PID**
First:	**P1**	**13**		First:		
Second:	**P3**	**27**		Second:		
Third:	**P2**	**5**		Third:		
Fourth:	**P5**	**10**		Fourth:		
Fifth:	**P4**	**3**		Fifth:		

Time: 7 msec. Process P2 runs.						
READY Queue				**WAIT Queue**		
Element	**PID**	**RBT**		**Element**	**trrq**	**PID**
First:	**P1**	**13**		First:	**27**	**P4**
Second:	**P3**	**27**		Second:		
Third:	**P2**	**5**		Third:		
Fourth:	**P5**	**10**		Fourth:		
Fifth:				Fifth:		

Time: msec. Process runs.						
READY Queue				**WAIT Queue**		
Element	**PID**	**RBT**		**Element**	**trrq**	**PID**
First:				First:		
Second:				Second:		
Third:				Third:		
Fourth:				Fourth:		
Fifth:				Fifth:		

Time: msec. Process runs.						
READY Queue				**WAIT Queue**		
Element	**PID**	**RBT**		**Element**	**trrq**	**PID**
First:				First:		
Second:				Second:		
Third:				Third:		
Fourth:				Fourth:		
Fifth:				Fifth:		

Answer: Process **P2** is selected to run **at 7 msec.**

Scheduling Algorithm (qualifier): **Round Robin (quantum = 12 msec)**

Time: 0 msec. Process P1 runs.						
READY Queue				**WAIT Queue**		
Element	**PID**	**RBT**		**Element**	**trrq**	**PID**
First:	**P1**	**14**		First:		
Second:				Second:		
Third:				Third:		
Fourth:				Fourth:		
Fifth:				Fifth:		

Time: 12 msec. Process P2 runs.						
READY Queue				**WAIT Queue**		
Element	**PID**	**RBT**		**Element**	**trrq**	**PID**
First:	**P2**	**7**		First:		
Second:	**P3**	**27**		Second:		
Third:	**P4**	**4**		Third:		
Fourth:	**P5**	**10**		Fourth:		
Fifth:	**P1**	**2**		Fifth:		

Time: 19 msec. Process P3 runs.						
READY Queue				**WAIT Queue**		
Element	**PID**	**RBT**		**Element**	**trrq**	**PID**
First:	**P3**	**27**		First:	**39**	**P2**
Second:	**P4**	**4**		Second:		
Third:	**P5**	**10**		Third:		
Fourth:	**P1**	**2**		Fourth:		
Fifth:				Fifth:		

Time: 31 msec. Process P4 runs.						
READY Queue				**WAIT Queue**		
Element	**PID**	**RBT**		**Element**	**trrq**	**PID**
First:	**P4**	**4**		First:	**39**	**P2**
Second:	**P5**	**10**		Second:		
Third:	**P1**	**2**		Third:		
Fourth:	**P3**	**15**		Fourth:		
Fifth:				Fifth:		

Time: 35 msec. Process P5 runs.						
READY Queue				**WAIT Queue**		
Element	**PID**	**RBT**		**Element**	**trrq**	**PID**
First:	**P5**	**10**		First:	**39**	**P2**
Second:	**P1**	**2**		Second:	**55**	**P4**
Third:	**P3**	**15**		Third:		
Fourth:				Fourth:		
Fifth:				Fifth:		

Time: 45 msec. Process P1 runs.						
READY Queue				**WAIT Queue**		
Element	**PID**	**RBT**		**Element**	**trrq**	**PID**
First:	**P1**	**2**		First:	**55**	**P4**
Second:	**P2**	**7**		Second:	**65**	**P5**
Third:	**P3**	**15**		Third:		
Fourth:				Fourth:		
Fifth:				Fifth:		

Time: msec. Process runs.						
READY Queue				**WAIT Queue**		
Element	**PID**	**RBT**		**Element**	**trrq**	**PID**
First:				First:		
Second:				Second:		
Third:				Third:		
Fourth:				Fourth:		
Fifth:				Fifth:		

Time: msec. Process runs.						
READY Queue				**WAIT Queue**		
Element	**PID**	**RBT**		**Element**	**trrq**	**PID**
First:				First:		
Second:				Second:		
Third:				Third:		
Fourth:				Fourth:		
Fifth:				Fifth:		

Answer: Process **P1** is selected to run **at 45 msec.**

Scheduling Algorithm (qualifier): **Priority Scheduling <u>without</u> Preemption**

Time: 0 msec. Process P1 runs.						
READY Queue				**WAIT Queue**		
Element	**PID**	**RBT**		**Element**	**trrq**	**PID**
First:	**P1**	**14**		First:		
Second:				Second:		
Third:				Third:		
Fourth:				Fourth:		
Fifth:				Fifth:		

Time: 14 msec. Process P5 runs.						
READY Queue				**WAIT Queue**		
Element	**PID**	**RBT**		**Element**	**trrq**	**PID**
First:	**P2**	**7**		First:	**34**	**P1**
Second:	**P3**	**27**		Second:		
Third:	**P4**	**4**		Third:		
Fourth:	**P5**	**10**		Fourth:		
Fifth:				Fifth:		

Time: 24 msec. Process P3 runs.						
READY Queue				**WAIT Queue**		
Element	**PID**	**RBT**		**Element**	**trrq**	**PID**
First:	**P2**	**7**		First:	**34**	**P1**
Second:	**P3**	**27**		Second:	**44**	**P5**
Third:	**P4**	**4**		Third:		
Fourth:				Fourth:		
Fifth:				Fifth:		

Time: 51 msec. Process P5 runs.						
READY Queue				**WAIT Queue**		
Element	**PID**	**RBT**		**Element**	**trrq**	**PID**
First:	**P2**	**7**		First:	**71**	**P3**
Second:	**P4**	**4**		Second:		
Third:	**P1**	**14**		Third:		
Fourth:	**P5**	**10**		Fourth:		
Fifth:				Fifth:		

Time: 61 msec. Process P4 runs.						
READY Queue				**WAIT Queue**		
Element	**PID**	**RBT**		**Element**	**trrq**	**PID**
First:	**P2**	**7**		First:	**71**	**P3**
Second:	**P4**	**4**		Second:	**81**	**P5**
Third:	**P1**	**14**		Third:		
Fourth:				Fourth:		
Fifth:				Fifth:		

Time: 65 msec. Process P1 runs.						
READY Queue				**WAIT Queue**		
Element	**PID**	**RBT**		**Element**	**trrq**	**PID**
First:	**P2**	**7**		First:	**71**	**P3**
Second:	**P1**	**14**		Second:	**81**	**P5**
Third:				Third:	**85**	**P4**
Fourth:				Fourth:		
Fifth:				Fifth:		

Time: msec. Process runs.						
READY Queue				**WAIT Queue**		
Element	**PID**	**RBT**		**Element**	**trrq**	**PID**
First:				First:		
Second:				Second:		
Third:				Third:		
Fourth:				Fourth:		
Fifth:				Fifth:		

Time: msec. Process runs.						
READY Queue				**WAIT Queue**		
Element	**PID**	**RBT**		**Element**	**trrq**	**PID**
First:				First:		
Second:				Second:		
Third:				Third:		
Fourth:				Fourth:		
Fifth:				Fifth:		

Answer: Process **P1** is selected **at 65 msec.**

Item xi: Consider the following sequence of ***Linux*** commands, and the indicated response shown to each. Each typed command is shown in bold, and the responses in plain typeface.

```
$ ls
demo        names       temp
$ mkdir /home/alex/literature
$ ls
demo        literature  names       temp
```

NOTE: From the two directory listings, we can conclude that the current working directory is "/home/alex"

```
$ ln names /home/alex/literature/newNamesOne
```

NOTE: This command results in the existence of two directory listings pointing to the inode of a single file. If the file is accessed via either one of the directory listings and is modified, then the changes will be seen by anyone accessing the file either from the same directory listing or the other directory listing.

```
$ cp temp /home/alex/literature/tempTwo
```

NOTE: This command creates a new file named "tempTwo" in the directory "/home/alex/literature". The new file is an exact copy of the file "temp" that had previously been present in "/home/alex". Since there are now two complete copies of the file, each file, "temp" and "tempTwo", can be modified without affecting the content of the other file.

```
$ rm names
```

NOTE: This command deletes the entry "names" from the directory /home/alex. However, the file remains on the disk and can still be accessed through the hard link newNamesOne that had previously been created in the directory /home/alex/literature

```
$ cd /home/alex/literature
$ ls
```

57. What response is obtained to the last command?

5 pts

Answer: **E**

A. demo names temp
B. demo literature names temp
C. newNamesOne temp tempTwo
D. newNamesOne temp
E. **newNamesOne tempTwo**
F. none of the above.

Item xii: If you had made this assignment: **$ furniture=table** indicate the output of each command.

2 pts each

58. **$ echo $furniture** *Answer*: **table**
59. **$ echo '$furniture'** *Answer*: **$furniture**
60. **$ echo "furniture"** *Answer*: **furniture**

Item xiii: If an *i-node* contains ten direct addresses of 4 bytes each, and all disk blocks are 1024 kB,

10 pts

61. what is the largest possible file?

Answer: **10,240 kB** or **10.24 MB**

Fall 2008 Mid-Term: CS-450 & CS-550

Question:	Answer:		Question:	Answer:		Question:	Answer:	
1	***F***	*2 pts*	26	***N***	*2 pts*	51	**P2**	*8 pts*
2	***T***	*2 pts*	27	***Y***	*2 pts*	52	**7 msec**	*8 pts*
3	***T***	*2 pts*	28	***Y***	*2 pts*	53	**P1**	*8 pts*
4	***F***	*2 pts*	29	***N***	*2 pts*	54	**45 msec**	*8 pts*
5	***T***	*2 pts*	30	***Y***	*2 pts*	55	**P1**	*8 pts*
6	***F***	*2 pts*	31	***Y***	*2 pts*	56	**65 msec**	*8 pts*
7	***F***	*2 pts*	32	***N***	*2 pts*	57	**E**	*5 pts*
8	***F***	*2 pts*	33	***C***	*2 pts*	58	**table**	*2 pts*
9	***F***	*2 pts*	34	***C***	*2 pts*	59	**$furniture**	*2 pts*
10	***F***	*2 pts*	35	***C***	*2 pts*	60	**furniture**	*2 pts*
11	***F***	*2 pts*	36	***B***	*2 pts*	61	**10.24 MB**	*10 pts*
12	***F***	*2 pts*	37	***A***	*2 pts*			
13	***N***	*2 pts*	38	***F***	*2 pts*			
14	***N***	*2 pts*	39	***F***	*2 pts*			
15	***N***	*2 pts*	40	***F***	*2 pts*			
16	***Y***	*2 pts*	41	***F***	*2 pts*			
17	***Y***	*2 pts*	42	***F***	*2 pts*			
18	***Y***	*2 pts*	43	***F***	*2 pts*			
19	***Y***	*2 pts*	44	***Y***	*2 pts*			
20	***Y***	*2 pts*	45	***Y***	*2 pts*			
21	***Y***	*2 pts*	46	***Y***	*2 pts*			
22	***Y***	*2 pts*	47	***Y***	*2 pts*			
23	***Y***	*2 pts*	48	**11**	*12 pts*			
24	***N***	*2 pts*	49	**14**	*12 pts*			
25	***Y***	*2 pts*	50	**13**	*12 pts*			