This exam is closed book, closed notes. All cell phones must be turned off. No calculators may be used.

You have two hours to complete this exam.

There are two parts to this exam: the first is true/false, the second is multiple choice. Some of the T/F questions are very simple, and some will take you awhile to determine. We expect you'll end up spending approximately equal amounts of time on each part.

Write all of your answers on the accu-scan form with a #2 pencil.

These exam questions must be returned at the end of the exam, but we will not grade anything in this booklet.

Unless stated (or implied) otherwise, you should make the following assumptions:
- The OS manages a single uniprocessor
- All memory is byte addressable
- The terminology \( \lg \) means \( \log_2 \)
- \( 2^{10} \) bytes = 1KB
- \( 2^{20} \) bytes = 1MB
- Page table entries require 4 bytes
- Data is allocated with optimal alignment, starting at the beginning of a page
- Assume leading zeros can be removed from numbers (e.g., 0x06 == 0x6).

Good luck!
Part 1: Straight-forward True/False [1 point each]
Designate if the statement is True (a) or False (b).

1) An operating system is defined as hardware that converts software into a useful form for applications.
   False. It is software that converts hardware.

2) Examples of resources that the OS must manage include CPU, memory, and disk.
   True.

3) The abstraction that the OS provides for the CPU is a virtual address space.
   False. The CPU abstraction is a process.

4) A process is defined as an execution stream (or thread of control) in the context of a process state.
   True.

5) The address space of a process is part of its process state.
   True.

6) A process is identical to a program.
   False; process is dynamic, program is static.

7) A process is identical to a thread.
   False; can have multiple threads in a single process.

8) Two processes reading from the same virtual address will access the same contents.
   False; each process has its own address space.

9) A modern OS virtualizes a single CPU with time-sharing.
   True.

10) Entering a system call involves changing from user mode to kernel mode.
    True.

11) When a user-level process wishes to call a function inside the kernel, it directly jumps to the desired function.
    False; must use system call; after generating a trap which is handled by the OS with a trap handler, the desired system call is invoked through the system call table.

12) An example of a mechanism inside the OS is the process dispatcher.
    True.

13) Cooperative multi-tasking requires hardware support for a timer interrupt.
    False; cooperative assumes process will voluntarily relinquish CPU or enter OS.

14) A timer tick is identical to a time slice.
    False; a time slice (how long a process is scheduled with RR) may be multiple timer ticks long.

15) PCB stands for process control base.
    False; Process Control Block.

16) On a uniprocessor system, there may only be one ready process at any point in time.
    False; just one RUNNING process, but many could be ready and want to be scheduled.

17) A FIFO scheduler schedules ready processes according to their arrival time.
    True.

18) The convoy effect occurs when high priority jobs must wait for lower priority jobs.
True/False both accepted; convoy effect usually implies short jobs must wait for long jobs, but one could consider a short job to be high priority and a long job to be low priority.

19) A **SJF scheduler** uses the past run time (i.e., cpu burst) of a job to predict future run time (i.e., cpu burst).
   False; SJF assumes an oracle with perfect knowledge of future behavior.

20) A **STCF scheduler** guarantees that it will schedule the ready job with the smallest remaining cpu burst.
   True.

21) A **RR scheduler** may preempt a previously running job.
   True.

22) An RR scheduler tends to decrease average response time as the time-slice is decreased.
   True; jobs will be scheduled sooner (decreasing response time) with a smaller time slice.

23) The shorter the time slice, the more a RR scheduler gives similar results to a FIFO scheduler.
   False; with a longer time slice (imagine the time slice > cpu burst), RR schedules the jobs without rotating between jobs.

24) If all jobs arrive at the same point in time, a SJF and an STCF scheduler will behave the same.
   True.

25) If all jobs have identical run lengths, a FIFO and a SJF scheduler will behave the same.
   True.

26) If all jobs have identical run lengths, a RR scheduler provides better average turnaround time than FIFO.
   False; if all jobs are identical, RR is horrible for turnaround time because all jobs will complete at nearly the same time.

27) A RR scheduler is guaranteed to provide the optimal average turnaround time for a workload.
   False; RR can provide horrible turnaround time (see example in 26).

28) A SJF scheduler requires an **oracle** to predict how long each job will perform I/O in the future.
   False; needs an oracle, but it is to predict the next CPU burst of each job.

29) With a **MLFQ scheduler**, compute-bound jobs are given higher priority.
   False; jobs that do a lot of computation (long CPU burst) are given low priority.

30) With a MLFQ scheduler, high priority jobs have longer time-slices than low priority jobs.
   False; high priorities have short time-slices so that interactive jobs can run promptly, but for just a short time.

31) With a MLFQ scheduler, jobs run to completion as long as there is not a higher priority job.
   False; at the lowest priority level, MLFQ will still RR across jobs of equal priority.

32) The OS provides the illusion to each process that it has its own address space.
   True.

33) The **static** portion of an address space cannot contain any data.
   False; read-only data could be in a static portion (along with code).

34) **Stacks** are used for procedure call frames, which include local variables and parameters.
   True.

35) Pointers should not reference the **heap**.
False; it can be bad practice to return pointers to data allocated on the stack.

36) An **instruction pointer register** is identical to a program counter.
   True.
37) A modern OS virtualizes memory with **time-sharing**.
   False, use space-sharing.
38) A virtual address is identical to a logical address.
   True.
39) With **dynamic relocation**, hardware dynamically translates an address on every memory access.
   True.
40) An **MMU** is identical to a **Memory Management Unit**.
   True.
41) The OS may not manipulate the contents of an MMU.
   False; OS sets up contents of MMU when switch to new process.
42) A disadvantage of **segmentation** is that different portions of an address space cannot grow independently.
   False; segmentation lets each segment grow independently.
43) A disadvantage of segmentation is that **segment tables** require a significant amount of space in memory.
   False; segment tables are usually small (just a base and bounds for each segment and not many segments).
44) With pure segmentation (and no other support), fetching and executing an instruction that performs a store from a register to memory will involve exactly one memory reference.
   False; fetching the instruction requires one memory reference and executing the store requires a second memory reference.
45) Paging approaches suffer from **internal fragmentation**, which grows as the size of a page grows.
   True.
46) A physical page is identical to a frame.
   True.
47) The size of a virtual page is always identical to the size of a physical page.
   True.
48) The number of virtual pages is always identical to the number of physical pages.
   False; the size of the virtual address space can be different than the amount of physical memory.
49) If 8 bits are used in a virtual address to designate an offset within a page, each page must be exactly **256 bytes**.
   True; \(2^8 = 256\) bytes.
50) If a physical address is 24 bits and each page is 4KB, the top 10 bits exactly designate the physical page number.
   False; \(\lg 4K = 12 \rightarrow 12\) bits for page offset; \(24 - 12 = 12\) bits for physical page number.
51) If a virtual address is 16 bits and each page is 128B, then each address space can contain up to 512 pages.
   True. \(\lg 128 = 7\) bits. \(16 - 7 = 9\) bits. \(2^9 = 512\) pages.
52) A linear **page table** efficiently maps physical page numbers to virtual page numbers.
   False; efficiently maps VPN to PPN (would be expensive to search other way through linear structure).
53) Given a fixed page size, the size of a linear page table increases with a larger address space.
   True; larger address space → more pages → more page table entries in linear table.

54) Given a constant number of bits in a virtual address, the size of a linear page table increases with larger pages.
   False; larger pages → fewer pages → fewer entries in linear page table.

55) Given a 28-bit virtual address and 1KB pages, each linear page table will consume $2^{18}$ bytes.
   False. 1KB pages -> 10 bit offsets. $28 - 10 = 18$ bits for VPN. Each PTE is 4 bytes (coversheet assumption).
   $2^{18} \times 4$ bytes.

56) Page table entries are stored in the PCB of a process when a context switch occurs.
   False; keep base of page table in PCB, but not individual page table entries.

57) Compared to pure segmentation, a linear page table doubles the required number of memory references.
   True; need to look up VPN->PPN in page table on every memory access.

58) A disadvantage of paging is that it is difficult to track free memory.
   False; pages are all same size so just use bitmap to track state (free vs. allocated) of each page.

59) A disadvantage of paging is that all pages within an address space must be allocated.
   False; each page table entry must be allocated (with a linear table), but if the page table entries shows the page
   isn’t valid, then the page doesn’t have to be allocated.

60) A TLB is identical to a Translation Lookahead Buffer.
   True or False; question thrown out (exam worth only 194 points instead of 196). In class, we only used term
   Translation Lookaside Buffer, but both terms are sometimes used.

61) A TLB caches translations from full virtual addresses to full physical addresses.
   False; TLB translates virtual page numbers to physical page numbers (no offset portion of address in TLB).

62) If a workload sequentially accesses 4096 4-byte integers stored on 256 byte pages, the TLB is likely to have a
   miss rate around $2^8$ (ignore any other memory references).
   False. Access 16KB of data sequentially; with 256 byte pages (and perfect alignment), this data fits on $2^4 / \ 2^6$ pages. Assume first access to each page misses in TLB, while remaining accesses to that page hit in TLB.
   Miss rate = # misses / # accesses = # pages / # accesses = $2^6 / 2^8 = 2^(-6).

63) If a workload sequentially accesses data, the TLB miss rate will decrease as the page size increases.
   True; with large pages, more accesses to same page, whose mapping will already be in TLB.

64) A workload that sequentially accesses data is likely to have good temporal locality, but not necessarily good 
   spatial locality.
   False; good spatial locality but not necessarily temporal.

65) TLB reach is defined as the number of TLB entries multiplied by the size of each TLB entry.
   False; TLB entries multiplied by the size of each page.

66) On a context switch, the TLB must be flushed to ensure that one process cannot access the memory of another
   process.
   False; can avoid flushing TLB if have ASIDs.

67) A longer scheduling time slice is likely to decrease the overall TLB miss rate in the system.
True; if a process is scheduled for a longer period of time, it will amortize the cost of the cold start misses to load up TLB with needed translations over a longer period of time.

68) On a **TLB miss**, the desired page must be fetched from disk.
   False; TLB miss means the page tables must be accessed; page fault will need to access disk.

69) With a TLB, only the outermost page table of each process needs to be accessed.
   False; if a TLB miss, still need to walk entire page table.

70) If the **valid bit** is clear (equals 0) in a PTE needed for a memory access, the running process is likely to be killed by the OS.
   True; if process tries to access page not valid in its address space, it is a segmentation fault.

71) There is a separate page table for every active process in the system.
   True.

72) An **inverted page table** is efficiently implemented in hardware.
   False; inverted page tables are implemented in software.

73) One advantage of adding segmentation to paging is that it potentially reduces the size of the page table.
   True; only need page table entries for valid pages in each separate segment.

74) One advantage of adding paging to segmentation is that it reduces the amount of internal fragmentation.
   False; Paging has internal fragmentation.

75) A single page can be **shared** across two address spaces by having each process use the same page table.
   False; if two processes use the same page table, all of their pages will be shared.

76) An advantage of a multi-level page table (compared to a linear page table) is that it potentially reduces the number of required memory accesses to translate an address.
   False; multiple levels increase the number of memory accesses needed for translation.

77) A **page directory** is identical to the outermost level of the page table.
   True.

78) With a multi-level page table, the complete VPN is used as an index into the page directory.
   False (but question mistakenly scored as True); only outer bits of VPN are used. Question thrown out and thus exam is worth 194 points instead of 196.

79) TLBs are more beneficial with multi-level page tables than with linear (single-level) page tables.
   True; if a TLB hit, able to avoid more memory lookups for page translation (higher cost of a miss).

80) Given a 2-level page table (and no TLB), exactly 2 memory accesses are needed to fetch an instruction.
   False; 2 accesses for 2 levels of address translation + 1 for fetch = 3 accesses.

81) With a multi-level page table, hardware must understand the format of PTEs.
   False; not necessary to have hardware support for multi-level page tables.

82) In the **memory hierarchy**, a backing store is faster than the memory layer above that uses that backing store.
   False; backing stores are larger and slower than the layers above that use it.

83) If the **present bit** is clear in a needed PTE, then the running process is likely to be killed by the OS.
   False; if present bit is clear, page must be brought in from disk.
84) A page fault is identical to a page miss.
    True.

85) When the dirty bit is set in a PTE, the contents of the TLB entry do not match the contents in the page table.
    False; dirty bit means contents of page in memory do not match contents on disk.

86) The OS can run a single process whose allocated address space exceeds the amount of physical memory available in the system.
    True.

87) The OS can run multiple processes whose total allocated address space exceeds the amount of physical memory available in the system.
    True.

88) When a page fault occurs, it is less expensive to replace a clean page than a dirty page.
    True; clean page can be simply discarded since it matches what is on disk; dirty page must be written to disk to update that (only) copy.

89) When a page fault occurs, the present bit of the victim page (i.e., the page chosen for replacement) will be cleared by the OS.
    True; the victim page will no longer be present in main memory.

90) A TLB miss is usually faster to handle than a page miss.
    True; TLB miss just requires accessing main memory; page miss requires accessing much slower disk.

91) Demand paging is identical to anticipatory paging.
    False; demand paging brings in page only when needed; anticipatory brings in pages beforehand.

92) Prefetching of pages helps sequential workloads to avoid page misses.
    True; prefetching works well with sequential accesses since can predict next page to be accessed.

93) LRU is an example of a mechanism for determining which page should be replaced from memory.
    False; LRU is a policy.

94) The OPT replacement policy replaces the page that is used the least often in the future.
    False; OPT replaces the page that will be used the furthest away in the future (not least often).

95) LRU always performs as well or better than FIFO.
    False; not necessarily.

96) OPT always performs as well or better than FIFO.
    True.

97) LRU with N+1 pages of memory always performs as well or better than LRU with N pages of memory.
    True, due to stack property.

98) FIFO with N+1 pages of memory always performs as well or better than FIFO with N pages of memory.
    False, see Belady’s anomaly.

99) LRU-K and 2Q use both how recently and how frequently a page has been accessed to determine which page should be replaced.
    True.
100) The **clock** policy replaces the least-recently-used page belonging to any process in the system.

   False; clock approximates LRU, but it doesn’t necessarily replace the single least-recently-used page.
Part 2: Multiple-Choice Questions [4 points each]
Assume three jobs arrive at approximately the same time, but Job A arrives slightly before Job B, and Job B arrives slightly before job C. Job A requires 2 sec of CPU, Job B is 8 secs, and Job C is 7 secs. Assume a time-slice of 1 sec.

101) Given a FIFO scheduler, what is the turnaround time of job B?
   a. 0 seconds
   b. 2 seconds
   c. 8 seconds
   d. 10 seconds
   e. None of the above
FIFO: A (until 2), B (until 10), C (until 17); B completes at time 10 seconds.

102) Given a FIFO scheduler, what is the average response time of the three jobs?
   a. 1 second
   b. 2 seconds
   c. 4 seconds
   d. 9.67 seconds
   e. None of the above
Average response time: \((0 + 2 + 10) / 3 = 4\) seconds

103) Given a RR scheduler, what is the turnaround time of job B?
   a. 1 second
   b. 4 seconds
   c. 16 seconds
   d. 17 seconds
   e. None of the above
B has the longest CPU burst and with RR will finish after every other job finishes, which is \(2 + 8 + 7 = 17\) seconds.
RR schedule: ABCABCBCBCBCBCB

104) Given a RR scheduler, what is the average response time of the three jobs?
   a. 1 second
   b. 2 seconds
   c. 3 seconds
   d. 12.33 seconds
   e. None of the above
Average response time: \((0 + 1 + 2) / 3 = 1\) second.

105) Given a SJF scheduler, what is the turnaround time of job B?
   a. 2 seconds
   b. 9 seconds
   c. 16 seconds
   d. 17 seconds
   e. None of the above
B is longest, so it is scheduled last; finishes when workload ends: \(2 + 7 + 8 = 17\).

106) Given a SJF scheduler, what is the average response time of the three jobs?
   a. 2 seconds
   b. 3.67 seconds
   c. 9 seconds
   d. 9.33 seconds
   e. None of the above
Average response time: \((0 + 2 + 9) / 3 = 3.67\)
Assume the OS schedules a workload containing three jobs with the following characteristics:

<table>
<thead>
<tr>
<th>Job</th>
<th>Arrival Time</th>
<th>CPU burst</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>B</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>C</td>
<td>12</td>
<td>2</td>
</tr>
</tbody>
</table>

107) Which scheduler minimizes the completion time of the entire workload?
   a. FIFO
   b. RR
   c. SJF
   d. STCF
   e. None of the above

All schedulers give the same result; the completion time for the entire workload (i.e., the last job) will be the same independent of the completion time of the individual jobs.

108) Given a STCF scheduler, what is the response time for job B?
   a. 5 seconds
   b. 7 seconds
   c. 10 seconds
   d. 20 seconds
   e. None of the above

B is scheduled at time 10 (at time 5 when B arrives, it has a burst of 8 > A’s remaining burst of 5); 10 -5 = 5.

109) Given a STCF scheduler, what is the response time for job C?
   a. 0 seconds
   b. 2 seconds
   c. 12 seconds
   d. 14 seconds
   e. None of the above

When C arrives at time 12, it has a CPU burst of 2 < B’s remaining CPU burst of 8-2=6; therefore, C is scheduled immediately when it arrives.

Assume you have an architecture with 1KB address spaces and 16KB of physical memory. Assume you are performing dynamic relocation with a base-and-bounds register. The base register contains 0x0000037d (decimal 893) and the bounds register contains 506 (decimal). Translate each of the following virtual addresses into physical addresses.

110) Virtual address 0x02e7 (decimal: 743) is physical address:
   a. 0x0000025d0 (decimal: 9680)
   b. 0x000002581 (decimal: 9601)
   c. 0x00003bc7 (decimal: 15303)
   d. Segmentation Violation
   e. None of the above

   743 > bounds of 506

111) Virtual address 0x01ef (decimal: 495) is physical address:
   a. 0x00000056c (decimal: 1388)
   b. 0x00000e93 (decimal: 3731)
   c. 0x00000d54 (decimal: 3412)
   d. Segmentation Violation
   e. None of the above

   495 + 893 = 1388

112) Virtual address: 0x01a0 (decimal: 416) is physical address:
   a. 0x00000c05 (decimal: 3077)
   b. 0x0000051d (decimal: 1309)
   c. 0x0000051e (decimal: 1310)
   d. Segmentation Violation
   e. None of the above

   416 + 893 = 1309
Assume dynamic relocation is performed with a **linear page table**. Assume the address space size is 16KB, phys mem size is 64KB, and page size is 256 bytes. In a PTE, the high-order bit is the VALID bit. If the bit is 1, the rest of the entry is the PFN. If the bit is 0, the page is not valid. The following are the contents of the page table (from entry 0 down to the max size):

```
0x80000007 0x80000051 0x00000000 0x00000000 0x800000d1 0x80000041 0x8000004d 0x80000018 0x80000051 0x00000000 0x00000000 0x800000a1 0x800000d1 0x00000000 0x00000000 0x800000c2 0x80000000 0x8000002c 0x80000015 0x8000004b 0x8000000f 0x8000006d 0x8000000f 0x80000038 0x80000030 0x800000ac 0x8000006f 0x8000002a 0x800000c7 0x800000e6 0x80000073 0x00000000 0x00000000 0x00000000 0x800000ea 0x800000e9 0x800000d0 0x800000f8 0x00000000 0x80000054 0x800000ce 0x8000000c 0x80000073 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x80000007 0x8000000d 0x8000006a 0x00000000 0x8000000f 0x80000039 0x80000007 0x8000000d 0x80000093 0x8000000c 0x8000000d 0x8000006a 0x80000040 0x80000039 0x80000007 0x8000000d 0x80000093 0x8000000c 0x8000000d 0x8000006a 0x80000040 0x80000007 0x8000000d 0x80000093 0x8000000c
```

113) Virtual Address 0x175e is physical address:
   a. 0x775e
   b. **0xc75e**
   c. 0xd85e
   d. Invalid
   e. None of the above

Page size = 256 bytes → 8 bits for offset
0x1753 -> page 0x17 (decimal 16+7=23)
Contents of PTE 23 = 0x800000c7 (valid)
PPN: c7
Physical address **0xc75e**

114) Virtual address 0x1940 is physical address:
   a. **0x7340**
   b. 0xa540
   c. 0x51940
   d. Invalid
   e. None of the above

0x1940 → page 0x19 (decimal 16 + 9 = 25)
Contents of PTE 25: 0x80000073 (valid)
PPN: 73
Physical address: **0x7340**

115) Virtual address 0x3b1e is physical address:
   a. 0x7b1e
   b. 0x0b1e
   c. 0x00e
   d. **Invalid**
   e. None of the above

0x3b1e → page 0x3b (decimal 3*16 + 11 = 59)
Contents of PTE 49: 0x00000000 (valid bit not set!!!)
116) When accessing virtual address 0x3457, what will be the first page accessed (decimal)?
   a. 3
   b. 51
   c. 54
   d. 64
   e. Error or None of the above
   Must first read page directory which we were told is stored in page 51 (pdbr = 51).

117) What contents (i.e., value) will be read on the first access (hexadecimal)?
   a. 0x0d
   b. 0x7f
   c. 0x92
   d. 0xaf
   e. Error or None of the above
   Address 0x3457 in binary is: 0110100011101.
   With 32 byte pages, least-significant 5 bits (bit0-bit4) are used for page offset.
   Bit10-bit14 are used for index into page directory: 01101 which is 8+4+1=13.
   The 13th entry of page 51 is 0x92.

118) What will be the second page accessed (hex)?
   a. 0x0a
   b. 0x0c
   c. 0x1e
   d. 0x92
   e. Error or None of the above
   0x92 in binary is 10010010. MSB of 1 means entry is valid. Rest of entry is 0x12. Next page table is stored on page 0x12, which is not one of the options...

119) What is the corresponding final physical address?
   a. 0x197
   b. 0x457
   c. 0x477
   d. 0x497
   e. Error or None of the above
   We access page 0x12 (decimal 18) and look at entry 00010 (middle 5 bits of virtual address) and get the contents 0x8c. 0x8c in binary is 10001100; MSB of 1 means entry is valid. Rest of entry is 0x0c, which is our PPN. Remember the offset is 1011. Therefore, final address is 01100 10111, which in hex is: 0x197.

120) What contents (i.e., value) will be read from the final physical address?
   a. 0x05
   b. 0x07
   c. 0x1c
   d. 0x1d
   e. Error or None of the above
   Go to physical page 0x0c (decimal 12) and look at offset 10111 (16+4+2+1 = 23). Contents are 0x1c.

121) When accessing Virtual Address 0x5830 what will be the first page accessed (decimal)?
   a. 51
   b. 54
   c. 58
   d. 59
   e. Error or None of the above
   Again, for any virtual address in this address space, we always start with the page directory, which is stored in page 51.
Assume the OS is performing page replacement on only 4 pages of physical memory. Assume the following access stream of virtual pages:

Access: 8
Access: 7
Access: 4
Access: 2
Access: 5
Access: 4
Access: 7
Access: 3
Access: 4
Access: 5
Access: 9
Access: 2
Access: 7
Access: 6
Access: 2
Access: 9
Access: 9

122) If the OS uses the OPT replacement policy, how many misses will it incur?
   a. 8
   b. 9
   c. 10
   d. 11
   e. None of the above

Contents of 4 pages of physical memory:
8 - miss
87 - miss
874 - miss
8742 - miss
5742 - miss
5742
5742
5342 - miss
5342
5942 - miss
5942
5942
7942 - miss
7962 - miss
7962
7962
7962

123) If the OS uses the FIFO replacement policy, how many misses will it incur?
   a. 8
   b. 9
   c. 10
   d. 11
   e. None of the above

Access: 8: 8 miss
Access: 7: 87 miss
Access: 4: 874 miss
Access: 2 : 8742 miss
Access: 5 : 5742 miss
Access: 4 : 5742
Access: 7 : 5742
Access: 3 : 5342 miss
Access: 4 : 5342
Access: 5 : 5342
Access: 9 : 5392 miss
Access: 5 : 5392
Access: 2 : 5392
Access: 7 : 5397: miss
Access: 6 : 6397 miss
Access: 2 : 6297 miss
Access: 9 : 6297
Access: 9 : 6297

f.
124) If the OS uses the LRU replacement policy, how many misses will it incur?
   a. 8
   b. 9
   c. 10
   d. 11
   e. None of the above

Access: 8 : 8 miss
Access: 7 : 87 miss
Access: 4: 874 miss
Access: 2: 8742 miss
Access: 5: 5742 miss
Access: 4: 5742
Access: 7: 5742
Access: 3: 5743 miss
Access: 4: 5743
Access: 5: 5743
Access: 9: 5943 miss
Access: 5: 5943
Access: 2: 5942 miss
Access: 7: 5972 miss
Access: 6: 5672 miss
Access: 2: 5672
Access: 9: 9672 miss
Access: 9: 9672

Congratulations on finishing a very long and detailed exam...