The exam was a 3-hour, closed-book, 1-sheet notes allowed, exam.

**Question 1: Warming up**

1a) Answer is fffffff00.
    
    255 \(^\land\) 255 = 0, 0 \mid 255 = 255, \sim(255) = \sim(0x000000ff) = 0xffffff00.
    
    (You could also expand out 255 as 0 \ldots 0000 1111 1111 and work out that 255 \(^\land\) 255 = 0.
    
    \(x^\land x\) is 0 for any \(x\).)

1b) “if” and “while”
    
    Other answers: “if” and “do”, “if” and “for”, “if” and “goto”.

1c) Neither.
    
    fp points to a FILE structure.

1d) Trap.

1e) There is no difference.

1f) Mathematical induction.

**Question 2: Potpourri**

Answers: T F F T F F T T F T F T F F T T F F T F T F

We considered an alternate answer correct in some cases, where the student provided a reasonable assumption.

**Question 3: Let me Snooze**

3a) setting alarm
    
    cancelling alarm

3b) setting alarm
    
    wakeup now, or hit snooze
    
    (3 times, due to alarm going off at 100, 102, 104 seconds.)

3c) setting alarm
    
    wakeup now, or hit snooze
    
    (2 times, due to alarm going off at 100, 102 seconds.)

    snoozing now …
3d) Add a line “snooze = 0;” after line 7, or 8, or 9, or 12.

**Question 4: Discover Fun in Assembly**

4a) Arguments to the recursive call of func are x, y-1.
4b) Base case for func is: y <= 0.
4c) Returned result for the base case is 1.
4d) The output is: result is 25.
4e) The output is: result is 1.
4f) Given x, y, func returns 1 if y is negative, and x to the power y otherwise.
4g) func would compute a wrong result when x and y are too big, such that an overflow occurs in the multiplication. Some students answered: when x=0 and y=0. Since 0^0 is regarded to be 1 mathematically, the computed result is correct in this case.
4h) The performance of func can be improved by:
   (i) Using registers instead of stack memory locations (in assembly). Registers can be accessed faster than memory, leading to performance improvement.
   (ii) Using a loop instead of recursion to compute exponents (in C). Calling functions has a lot of overhead (pushing/popping instruction register, arguments etc. on stack), whereas a loop avoids that overhead.
   Other answers include: function inlining or loop unrolling, but only after converting to a loop first.

**Question 5: Store Virtually**

5a) Page size is 32 KB = 2^15 bytes. Therefore, offset is the last 15 bits, and virtual page number is the first 49 bits in the 64-bit virtual address. You have to re-arrange the binary digits into groups of 4. Thus, the “9” in the virtual address splits, giving 3FF concatenated with 1 for the page number, and 001 concatenated with ABC for the offset.

Virtual page number: 0x7FF
Offset in page: 0x1ABC

5b) Physical address has 32 bits since physical memory is 4 GB = (2^2 * 2^30) bytes. The mapped physical address has the same offset in the last 15 bits, and the first 17 bits represent 6 in hexadecimal.

Physical address is: 0x0003 1ABC
5c) Page fault
    Segmentation fault

5d) Spatial locality means that if a program references memory location X now, it will probably reference locations in storage nearby X soon. This applies to locations that store instructions and locations that store data. Caches act as a staging area for slower devices (lower in the storage hierarchy) to provide faster access. Therefore, programs that have spatial locality will likely get cache hits (i.e., find the instructions and data they need in the caches) leading to performance improvement. (Aside: Modern processors keep separate caches for instructions and data.)

**Question 6: Campaign Processes**

6a) Obama is done. Bernie is the new President!
    Obama is done. Trump is the new President!
    Obama is done. Hillary is the new President!

6b) fflush(stdout);

6c) Obama is done. Bernie is the new President!
    is done. Trump is the new President!
    is done. Hillary is the new President!

6d) The three sets of possible solutions are:

1. B1: uncommented and blanks filled with exit(0);
   B2: uncommented and blanks filled with exit(0);
   B3: commented out, or uncommented and blanks filled with fflush(stdout);

2. B1: uncommented and blanks filled with exit(0);
   B2: commented out, or uncommented and blanks filled with fflush(stdout);
   B3: uncommented and blanks filled with exit(0);

3. B1: commented out, or uncommented and blanks filled with fflush(stdout);
   B2: uncommented and blanks filled with exit(0);
   B3: uncommented and blanks filled with exit(0);

6e) The corresponding correct and complete outputs are:

1. Obama is done. Bernie is done. Trump is done. Hillary is the new President!

2. Obama is done. Bernie is done. Trump is the new President!
   is done. Hillary

3. Obama is done. Bernie is the new President!
   is done. Trump is done. Hillary
Question 7: “Like” Modularity

#ifndef ACTORDATA_INCLUDED
#define ACTORDATA_INCLUDED

/* An ActorData object is a structure that includes:
   the name of an actor,
   the number of likes received by the actor in colleges, and
   the total number of likes. */
typedef struct ActorData *ActorData_T;

/* Return a new ActorData_T object for actor with name *pcName,
   or NULL if insufficient memory is available. */
ActorData_T ActorData_new(const char *pcName);

/* Free oActorData. Do nothing if oActorData is NULL. */
void ActorData_free(ActorData_T oActorData);

/* Return the name in oActorData, or NULL if oActorData is NULL. */
char *ActorData_getName(ActorData_T oActorData);

/* Increment the Like number in oActorData for college iCollege */
void ActorData_incrementLike(ActorData_T oActorData, int iCollege);

/* Print the data in oActorData */
void ActorData_print(ActorData_T oActorData);

#endif