Big Picture – CPU, Memory, and Pages:

Page Eviction/Replacement Strategies:
When we need to remove a page from RAM and store it on disk, how do we decide which page to remove given a page access pattern?

Strategy #1:

```
   17   33   40   17   43   8   99   33   99   17
RAM
```

Strategy #2:

```
   17   33   40   17   43   8   99   33   99   17
RAM
```

Strategy #3:

```
   17   33   40   17   43   8   99   33   99   17
RAM
```

Strategy #4:

```
   17   33   40   17   43   8   99   33   99   17
RAM
```

Other Strategies:

Fragmentation
As we develop various systems for storage, we want to minimize fragmentation.

- [Fragmentation]:
- [Internal Fragmentation]:
- [External Fragmentation]:

Fragmentation Example in Heap Memory:

```
Unallocated (3072 bytes) ⇐ End of Heap
Used (4096 bytes)
Free (3072 bytes)
Used Data (2048 bytes) ⇐ Start of Heap
```
Computer Peripherals

- Every other piece of hardware we consider to be a “peripheral”.
- Interface managed by the ________________________.
  - ...and managed using _____________________.
- Examples:

Threads: The Unit of Computation in an Operating System
As a programmer, the single most important construct in an Operating System is a thread.

- Every thread has a program counter, a pointer that stores the next instruction to be read by a program.
- A ______________ is an organization of one or more threads in the same context. A simple process has only one thread.
- In C, the initial thread is called the __________________.
  - It is what starts running your main() function!

Creating Additional Threads in C
The pthread library is the POSIX thread library allowing you to create additional threads beyond the main thread.

Creating a new thread is a complex call with four arguments:

```c
int pthread_create(
    pthread_t *thread, /* thread struct */
    const pthread_attr_t *attr, /* usually NULL */
    void *(*start_routine) (void *), /* start func */
    void *arg /* thread start arg */
);
```

The start_routine has a very interesting type signature:
```c
void *(*start_routine) (void *)
```
This signature is a function pointer (“functor”) and is the syntax we can use to pass a pointer to a function. Therefore, the third argument into pthread_create must be a function with the following prototype:
```c
void *__________(void *ptr);
```
...you can use any name for the function name.

Example: Launching Fifteen Threads

```c
#include <pthread.h>

const int num_threads = 15;

void *thread_start(void *ptr) {
    int id = *((int *)ptr);
    printf("Thread %d running...
", id);
    return NULL;
}

int main(int argc, char *argv[]) {
    // Create threads:
    int i;
    pthread_t tid[num_threads];
    for (i = 0; i < num_threads; i++) {
        pthread_create(&tid[i], NULL,
                        thread_start, (void *)&i);
    }
    printf("Done!\n");
    return 0;
}
```

Q1: What is the expected output of this program?

Q2: What actually happens?

Q3: What do we know about threads in C?