1) Consider a logical address space of 64 pages of 1,024 words each, mapped onto a physical memory of 32 frames.

   a. How many bits are there in the logical address?

   b. How many bits are there in the physical address?

2) Assuming a 1-KB page size, what are the page numbers and offsets for the following address references (provided as decimal numbers):

   a. 3085

   b. 42095

   c. 215201

   d. 650000

   e. 2000001

3) The BTV operating system has a 21-bit virtual address, yet on certain embedded devices, it has only a 16-bit physical address. It also has a 2-KB page size. How many entries are there in each of the following?

   a. A conventional, single-level page table

   b. An inverted page table

   What is the maximum amount of physical memory in the BTV operating system?

4) Consider a logical address space of 256 pages with a 4-KB page size, mapped onto a physical memory of 64 frames.

   a. How many bits are required in the logical address?

   b. How many bits are required in the physical address?

5) Consider a computer system with a 32-bit logical address and 4-KB page size. The system supports up to 512 MB of physical memory. How many entries are there in each of the following?

   a. A conventional, single-level page table

   b. An inverted page table

6) The MPV operating system is designed for embedded systems and has a 24-bit virtual address, a 20-bit physical address, and a 4-KB page size. How many entries are there in each of the following?

   a. A conventional, single-level page table

   b. An inverted page table

   What is the maximum amount of physical memory in the MPV operating system?

7) Consider a logical address space of 2,048 pages with a 4-KB page size, mapped onto a physical memory of 512 frames.

   a. How many bits are required in the logical address?

   b. How many bits are required in the physical address?

8) Assume that a system has a 32-bit virtual address with a 4-KB page size. Write a C program that is passed a virtual address (in decimal) on the command line and have it output the page number and offset for the given address. As an example, your program would run as follows:

```
./addresses 19986
```

   Your program would output:

```
The address 19986 contains:
page number = 4
offset = 3602
```

Programming Project: Contiguous Memory Allocation

This project will involve managing a contiguous region of memory of size MAX where addresses may range from 0 ... MAX − 1. Your program must respond to four different requests:

1. Request for a contiguous block of memory
2. Release of a contiguous block of memory
3. Compact unused holes of memory into one single block
4. Report the regions of free and allocated memory

Your program will be passed the initial amount of memory at startup. For example, the following initializes the program with 1 MB (1,048,576 bytes) of memory:

```
./allocator 1048576
```

Once your program has started, it will present the user with the following prompt:

```
allocator>
```

It will then respond to the following commands: `RQ` (request), `RL` (release), `C` (compact), `STAT` (status report), and `X` (exit).

A request for 40,000 bytes will appear as follows:

```
allocator>RQ P0 40000 W
```

The first parameter to the `RQ` command is the new process that requires the memory, followed by the amount of memory being requested, and finally the strategy. (In this situation, "`W`" refers to worst fit.)

Similarly, a release will appear as:

```
allocator>RL P0
```

This command will release the memory that has been allocated to process `P0`.

The command for compaction is entered as:

```
allocator>C
```

This command will compact unused holes of memory into one region.

Finally, the `STAT` command for reporting the status of memory is entered as:

```
allocator>STAT
```

Given this command, your program will report the regions of memory that are allocated and the regions that are unused. For example, one possible arrangement of memory allocation would be as follows:

```
Addresses [0:315000] Process P1
Addresses [315001: 512500] Process P3
Addresses [512501:625575] Unused
Addresses [625575:725100] Process P6
Addresses [725001] . . .
```