1. Suppose a computer contains 512MB of main memory, and an operating system needs to create a virtual memory of twice the size of main memory and uses a paging system with page size of 2KB. How many pages would be created?

   Since the operating system needs twice size of main memory size, main memory size = 512MB = $2^9 \times 2^{20}$Byte = 2^{29}Byte, virtual memory size = main memory size = 512MB$\times$ 2 = 2^{30}Byte, and page size = 2KB = 2^{11}Byte.

   So, the number of pages we need is $\frac{2^{30}\text{Byte}}{2^{11}\text{Byte}} = 2^{19}$ pages.

2. A banker has $100,000 and loans $50,000 to each of two customers. Later, both customers return with the story that before they can repay their loans they must each borrow another $10,000 to complete the business deals in which their previous loans are involved. The banker resolves this deadlock by borrowing the additional funds from another source and passing on the loan (with an increase in the interest rate) to the two customers. Which of the ~~three~~ four conditions for deadlock has the banker removed?
   He resolves condition 1, Competition for non-sharable resources.

   註解：

   Remove 第一個 condition 最主要的就是"增加 resource"或是把 resource 變成 "shareable".

   Remove 第二個 condition 最主要的就是"不允許一次要一些",要 resource 就一次全要齊。就這個例子這個 banker 沒辦法用 remove 第二個 condition 來解決已經發生的問題。

   Remove 第三個 condition 最主要的就是"要回已經給出去的 resource"。以這個例子而言，banker 要強制其中一個人先還錢，再把錢借給另一個。

   Remove 第四個 condition 最主要的就是"resouce 給的順序要一致"。在這個例子，banker 沒辦法用 remove 第四個 condition 來解決已經發生的問題。但是 question 3 示範了一個例子是 Remove 第四個 condition。

3. Suppose each non-shareable resource in a computer system is classified as a level 1, level 2, or level 3 resource. Moreover, suppose each process in the system is required to request the resources it needs according to this classification. That is, it must request all the required level 1 resources at once before requesting any level 2 resources. Once it receives the level 1 resource, it can request all the required level 2 resources, and so on. Can deadlock occur in

such a system? Why or why not?

Deadlock **cannot** occur in such a system. For the resources in the same level, a process requests all it needed, which removes the 2nd condition of deadlock. For resources in different levels, processes need to retrieve them in order, which removes the 4th condition of deadlock. Therefore, no deadlock can occur.

4. A *barrier* for a group of threads/processes in the source code means any thread/process must stop at this point and cannot proceed until all other threads/processes reach this barrier. The following code uses a global variable `count`, whose initial value is 8, to implement a barrier for 8 threads/processes. (1) Give an example to show this code could fail in a single CPU multitasking environment. (2) How to fix this code to make it work? Justify your answers.

```
count = count – 1;
while (count > 0);
// barrier point
```

(1)
If process1 accesses *count = 8* at T1, then do *count - 1*. Then if process2 accesses *count* at this moment before process 1 sets *count* value as 7, so process 2 gets *count* equal to 8, too. After process1 and process2 finish the instruction *count = count -1*; ***count*** is 7. And both process1 and process 2 are waiting in the while loop. When all other processes finish the first instruction, **count** is still 1. This means every process is still hanging in the while loop, and cannot reach the barrier point forever.
(2)
The instruction `count = count -1;` is the critical region of this section of code. We just need to make sure the mutual exclusion of this instruction. We can use `disable_interrupt()` or `test-and-set` to reach our goal. For example, we can modify the code as

```
Disable_interrupt();
count = count – 1; //critical region
Enable_interrupt();
while (count > 0);
// barrier point
```