报错：

```
g++ print.cpp -o print
./print
malloc(): corrupted top size
make: *** [Makefile:3: all] Aborted (core dumped)
```

malloc(): corrupted top size 表示在调用 malloc()（动态内存分配）时，堆内存的 "top size" 被破坏了。

"corrupted" 指的是内存堆的结构出了问题，不过好像不是老师说的字段错误?

报错研究完了，再研究下代码：

首先：

```
int matrix[array_number][array_number];
```

这段代码分配了一个连续的空间，我们管他叫A矩阵吧。

在开局运行的函数里：

```
int **result = new int*[8];
```

这里分配了一个指针数组 `result`，它包含 8 个 `int*` 元素，在内存上说，应该是八个连续的内存块。每个 `int*` 将指向一个整数数组。

然后下面是一个for循环，给矩阵的每一排赋值，然后赋64次。每赋值完一排，就把一个`int`指针指向这一排（也就是A矩阵里对应的地址）。

其实分析到这里，错误就已经很明显了。因为只分配了8个指针，而矩阵有64个行，很明显这里会溢出。所以修改也只需要把8改成64即可。

所以不cout的话，修改这个代码很简单。

不过诡异的是，为什么cout了之后，这个东西反而能跑了?

我在gdb里面break malloc跟踪了一下内存的分配：

```
(gdb) continue
Continuing.

Breakpoint 1, __GI___libc_malloc (bytes=1024) at malloc.c:3023
3023    in malloc.c
```

开始分配内存了， backtrace一下。

```
(gdb) backtrace
#9  0x00005555555552f3 in main () at print.cpp:23
```

print.cpp:23，是cout那一行。再用finish返回一下：

```
(gdb) finish
Run till exit from #0  __GI___libc_malloc (bytes=1024) at malloc.c:3023
0x00007ffff7c68d04 in __GI__IO_file_doallocate (fp=0x7ffff7dd46a0
<_IO_2_1_stdout_>) at filedoalloc.c:101
101     filedoalloc.c: No such file or directory.
Value returned is $2 = (void *) 0x55555556eeb0
```

这里应该是分配了一个缓冲区，因为finish之后他返回了0x55555556eeb0，他又说是1024个字节，所以我猜缓冲区应该是从0x55555556eeb0开始到0x55555556f2b0结束。然后继续调试：

```
(gdb) continue
Continuing.
A magic print! If you comment this, the program will break.
```

这里成功cout了，应该是cout函数里的malloc被我们截断了。

下面又有代码申请了64字节，我们来看一下：

```
(gdb) backtrace
#0  __GI___libc_malloc (bytes=64) at malloc.c:3023
#1  0x00007ffff7e83b29 in operator new(unsigned long) () from /lib/x86_64-linux-
gnu/libstdc++.so.6
#2  0x0000555555555243 in double_array (n=64) at print.cpp:10
#3  0x0000555555555312 in main () at print.cpp:24
(gdb) finish
Run till exit from #0  __GI___libc_malloc (bytes=64) at malloc.c:3023
0x00007ffff7e83b29 in operator new(unsigned long) () from /lib/x86_64-linux-
gnu/libstdc++.so.6
Value returned is $3 = (void *) 0x55555556f2c0
```

按理来说`int ***result = new int*[64];`分配的内存将从`0x55555556f2c0`开始，直到`0x55555556f2c0 + 512 - 1`，即`0x55555556f2ff`。但是注意到，`0x55555556f2b0`（cout缓冲区结束）是比`0x55555556f2c0`（result开始分配）低的，所以这也是很诡异的一个点。如果说malloc这里申请64个字节的时候，是从低位到高位的，那么就解释不了这个东西，这里卡了我很久，最后我猜，这里malloc如果从高位到低位分配空间，就可以证明了。