笔试题：

1.Test（）执行会输出什么结果（返回栈空间地址的问题）？ ------ 出现非法访问内存现象

```
char* GetMemory()
{
 char p[] = "hello world"; //如果用static修饰 则正确 此时变量处于静态区
 return p;
} //当函数调用完之后 虽然将p的地址返回给了str
//但是 此时该地址的内容已经被释放 还给计算机 那么此时其中存放的内容就是不确定的

void Test(void)
{
  char* str = NULL;
  str = GetMemory();
  printf(str);
 }

 int main()
 {
  Test();
  return 0;
 }
```

2.代码有什么错误?

```
void GetMemory(char **p, int num)
{
 *p = (char*)malloc(num);
}

void Test(void)
{
 char* str = NULL;
 GetMemory(&str, 100);
  strcpy(str, "hello");
  printf(str);
  //此处忘记释放动态开辟的内存 导致内存泄漏
 }

```

```c
int main()
{
 Test();
 return 0;
}
```

3.结果是什么 有什么问题？----- 非法访问内存

```c
void test(void)
{
 char * str = (char*)malloc(100);
 strcpy(str, "hello");
 free(str); //free 释放str指向的空间后 并不会把str置为NULL 需要主动置为空指针
 if(str != NULL) //已经释放了空间 再次使用
 {
 strcpy(str, "world");
 printf(str);
  }
}

int main()
{
 test();
 return 0;
}
```

柔性数组

> 结构体中最后一个元素是未知大小的数组 那么这个数组称为柔性数组成员 大小是可以调整的
> 优点：
> 1.方便内存释放
> 2.有利于访问速度
> 特点：
> 1.结构体柔性数组成员前面至少包含一个成员
> 2.sizeof 返回的这种结构的大小不包含柔性数组的内存
> 3.包含柔性数组成员的结构体用malloc函数进行内存的动态分配，并且分配的内存应该大于结构体的大小 以适应柔性数组的预期大小

```c
struct S
{
 int n;
 int arr[0]; //柔性数组
```

```c
}

int main()
{
 struct S* s = (struct S*)malloc(sizeof(struct S) + 5*sizeof(int));
 //前四个空间是 int n 其余的是开辟的arr的内存空间
 return 0;
}
```