编号：

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 实验 | 一 | 二 | 三 | 四 | 五 | 六 | 七 | 八 | 九 | 十 | 总评 | 教师签名 |
| 成绩 |  |  |  |  |  |  |  |  |  |  |  |  |

武汉大学计算机学院

课程实验(设计)报告

专业(班)： 计算机科学与技术1班

学 号： 2015301040209

姓 名： 张相宾

课程名称： 操作系统设计

任课教师： 宋伟

2017年 11 月 18 日

**主存空间的分配和回收**

1. **实习题目**
2. 实习内容

主存储器空间的分配和回收。

1. 实习目的

通过本实习帮助理解在不同的存储管理方式下应怎样进行存储空间的分配和回收。

1. 实习题目

第一题：

可变分区管理方式下采用首次适应算法实现主存分配和回收

(1) 可变分区方式是按作业需要的主存空间大小来分割分区的。当要装入一个作业时，根据作业需要的主存容量查看是否有足够的空闲空间，若有，则按需分配，否则，作业无法装入。假定内存大小为128K(可输入)，空闲区说明表格式为：

·起始地址——指出空闲区的起始地址；

·长度——一个连续空闲区的长度；

·状态——有两种状态，一种是“未分配”状态；另一种是“空表目”状态，表示该表项目前没有使用。

(2) 采用首次适应算法分配回收内存空间。运行时，输入一系列分配请求和回收请求。

要求能接受来自键盘的空间申请及释放请求，能显示分区分配及回收后的内存布局情况。

1. **实习内容及设计思想**
2. 设计思路

a.分区分配算法：

为了将一个作业装入内存，应按照一定的分配算法从空闲分区表中选出一个满足作业需求的分区分配个作业，如果这个空闲分区的容量比作业申请的空间容量要大，则将该分区一分为二，一部分分配给作业，剩下的一部分仍然留在空闲分区表中，同时还要对空闲分区表中的有关信息进行修改。‘

本实验采用首次适应算法，空闲分区按地址递增的次序排列。在进行内存分配时，从空闲分区表首开始顺序查找，直到找到第一个能满足其大小要求的空闲分区为止。然后，再按照作业大小，从该分区中划出一块内存空间分配给请求者，余下的空闲分区仍然留在空闲分区表中。

b.分区的回收：

当作业执行结束时，应回收已使用完毕的分区。系统根据回收分区的大小及起始地址，在空闲分区表中检查是否有与其相邻的空闲分区，若有相邻分区，则应合并成为一个大的空闲区，然后修改有关的分区状态信息。回收分区与已有空闲分区的相邻情况有以下四种：

（1）回收分区r上邻接一个空闲分区，此时应将回收区r与上邻接分区F1合并成一个连续的空闲分区。合并分区的首地址为空闲分区F1的首地址，其大小为二者之和。

（2）回收分区r下邻接一个空闲分区，此时应将回收区r与下邻接分区F2合并成一个连续的空闲分区。合并分区的首地址为回收分区r的首地址，其大小为二者之和。

（3）回收分区r上、下邻接空闲分区，此时应将回收区r与上、下邻接分区合并成一个连续的空闲分区。合并分区的首地址为与r上邻接空闲分区F1的首地址，其大小为三者之和，且应把与r下邻接的空闲分区F2从空闲分区表中删去。

（4）回收分区r不与任何空闲分区相邻，这时应为回收区单独建立一个新表项，填写分区大小及其实地址等信息，并将其加入到空闲分区表中的适当位置。

1. 主要数据结构

Class area{

int size; //分区大小

int address;//分区起始地址

bool state; //1为已分配，0为未分配

int jobId; //分区对应的作业号

}

1. 主要代码结构

a.分区分配算法代码

iter = mem.begin();

while (iter != mem.end()) {

if (iter->size > size&&iter->state == false) {

int s = iter->size - size;

int a = iter->address + size;

iter->jobId = jobId;

iter->size = size;

iter->state = true;

iter++;

mem.insert(iter, area(s, a, false, -1));

judge = true;

break;

}

else if (iter->size == size&&iter->state == false) {

iter->jobId = jobId;

iter->state = true;

judge = true;

break;

}

else

iter++;

}

b.分区回收算法代码

iter = mem.begin();

while (iter != mem.end()) {

if (iter->jobId == jobId) {

fiter = iter - 1;

biter = iter + 1;

if (iter == mem.begin()) {

if (biter->state == false) {

size = biter->size + iter->size;

iter->size = size;

iter->state = false;

iter->jobId = -1;

iter++;

mem.erase(iter);

}

else {

iter->state = false;

iter->jobId = -1;

}

judge = true;

break;

}

if (biter == mem.end()) {

if (fiter->state == false) {

size = fiter->size + iter->size;

fiter->size = size;

mem.erase(iter);

}

else {

iter->state = false;

iter->jobId = -1;

}

judge = true;

break;

}

if (fiter->state == false && biter->state == false) {

size = fiter->size + iter->size + biter->size;

fiter->size = size;

iter = mem.erase(iter);

mem.erase(iter);

}

else if (fiter->state == false) {

size = fiter->size + iter->size;

fiter->size = size;

mem.erase(iter);

}

else if (biter->state == false) {

size = biter->size + iter->size;

iter->size = size;

iter->state = false;

iter->jobId = -1;

iter++;

mem.erase(iter);

}

else {

iter->state = false;

iter->jobId = -1;

}

judge = true;

break;

}

else {

iter++;

}

}

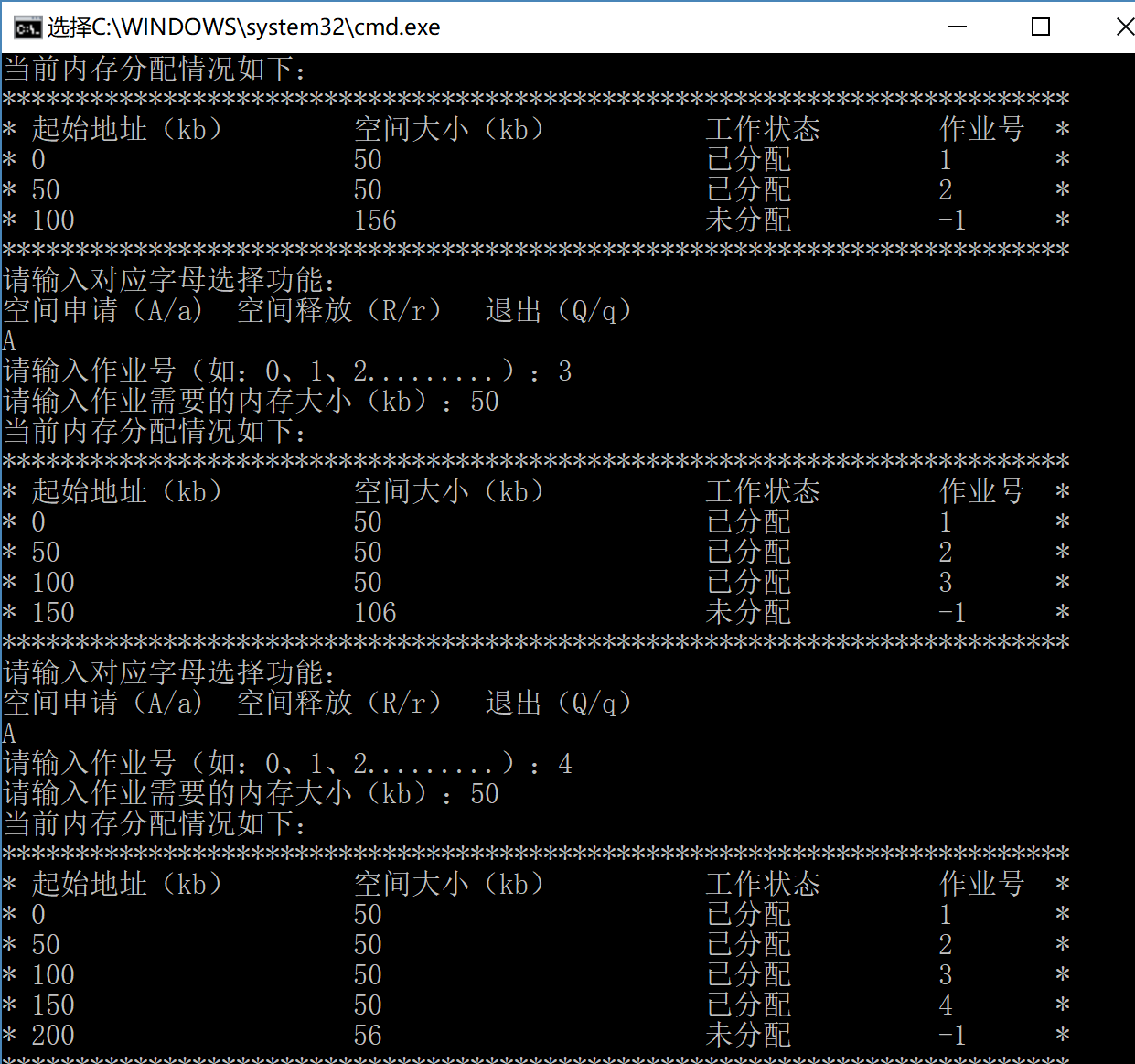
1. 代码段分析
2. **上机实习所用平台及相关软件**
3. **调试过程**
4. 测试数据设计

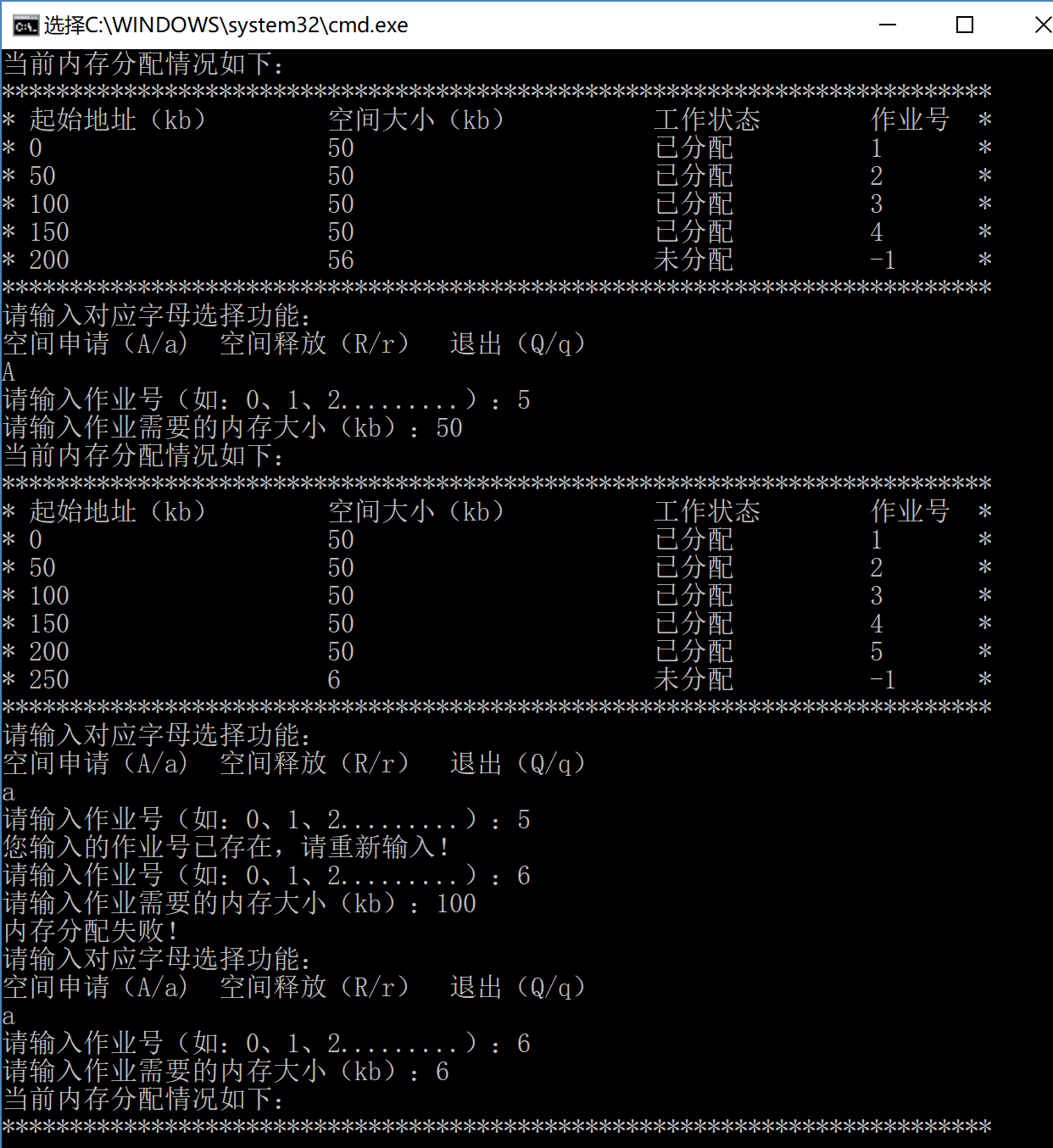
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序列 | 进程号 | 大小 | 执行 | 预测结果 |
| 1 | 1 | 50 | 申请空间 | 成功 |
| 2 | 2 | 50 | 申请空间 | 成功 |
| 3 | 3 | 50 | 申请空间 | 成功 |
| 4 | 4 | 50 | 申请空间 | 成功 |
| 5 | 5 | 50 | 申请空间 | 成功 |
| 6 | 5 | 1 | 申请空间 | 进程已存在，无法分配 |
| 7 | 6 | 100 | 申请空间 | 剩余空间不足，无法分配 |
| 8 | 6 | 6 | 申请空间 | 成功 |
| 9 | 3 | ~ | 回收进程 | 前分后分 |
| 10 | 5 | ~ | 回收进程 | 前分后分 |
| 11 | 4 | ~ | 回收进程 | 前空后空 |
| 12 | 6 | ~ | 回收进程 | 前空（最后一片空间） |
| 13 | 2 | ~ | 回收进程 | 后空前分 |
| 14 | 1 | ~ | 回收进程 | 后空（第一片空间） |

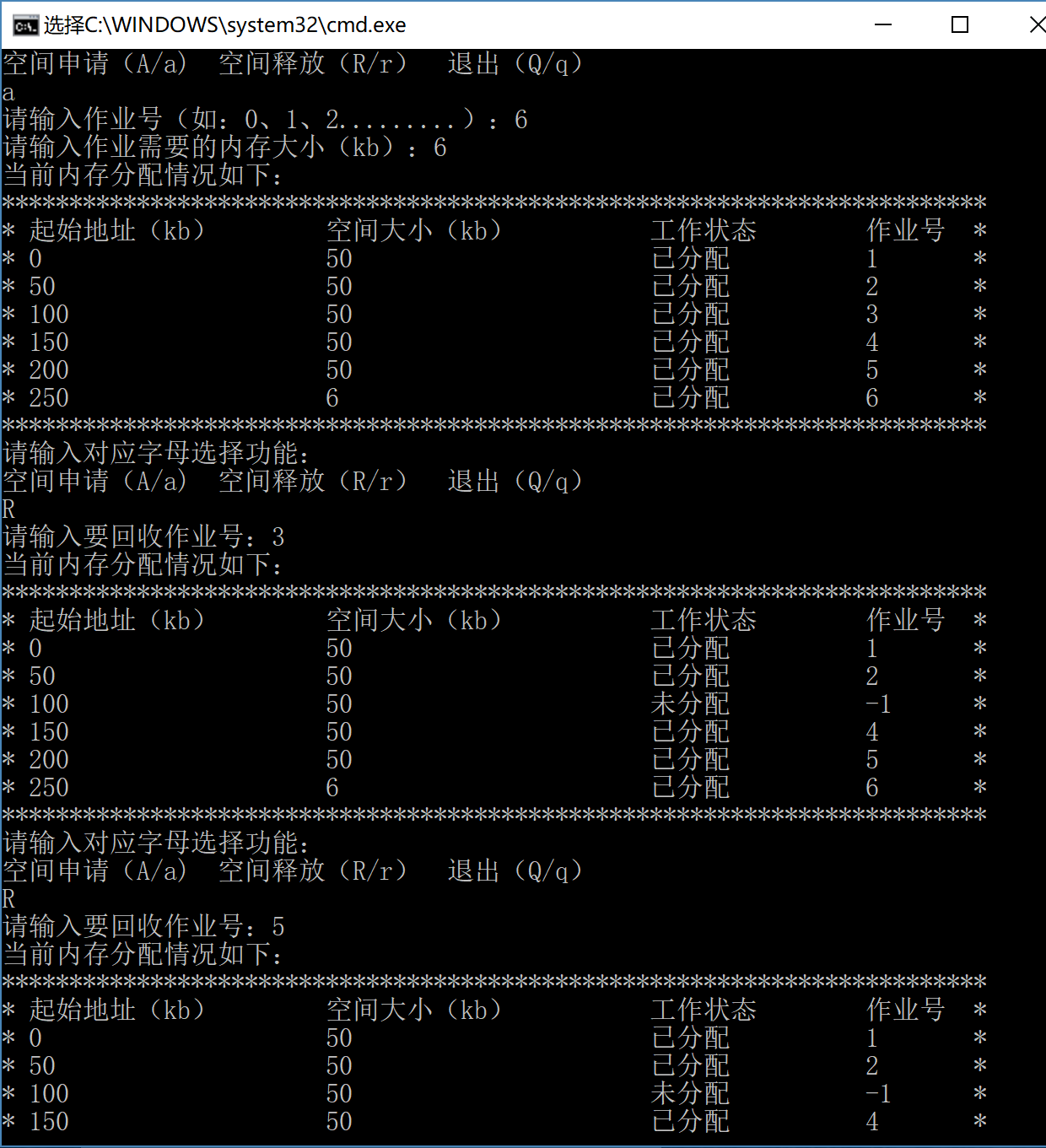
验证所有可能情况

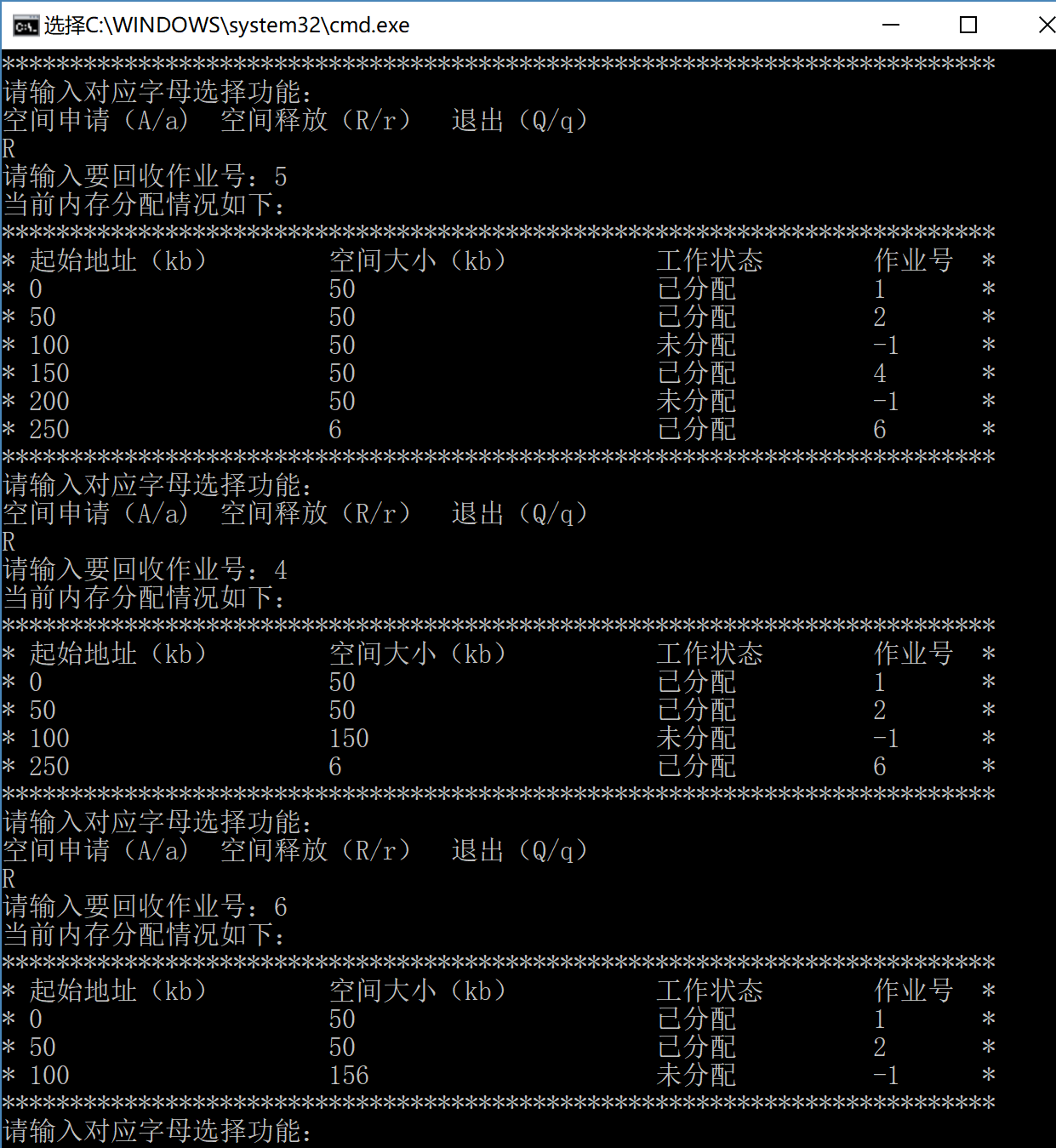
1. 测试结果分析













1. **总结**
2. 实习中遇到的问题及解决方法
   1. 问题：用户一般的输入错误，尤其是cin的理解

解决方法：输入要加上判断语句并有所提示，如选择功能时输入错误重新选择

* 1. 问题：开始没有考虑到用户可能会添加已存在的作业号，或者删除不存在的作业号，如果不对进程序列进行遍历查找是否存在，则会出现错误；

解决方法：每次先判断是否存在，再根据情况给出对应提示

* 1. 问题：最初没有考虑内存头和尾的回收，导致错误

解决方法：添加新的情况判断

1. 实习体会及收获

对首次适应算法和内存中进程空间的分配与回收有了更深的理解，并且对于c++迭代器和cin的使用进一步加深，对情况讨论也有了提高