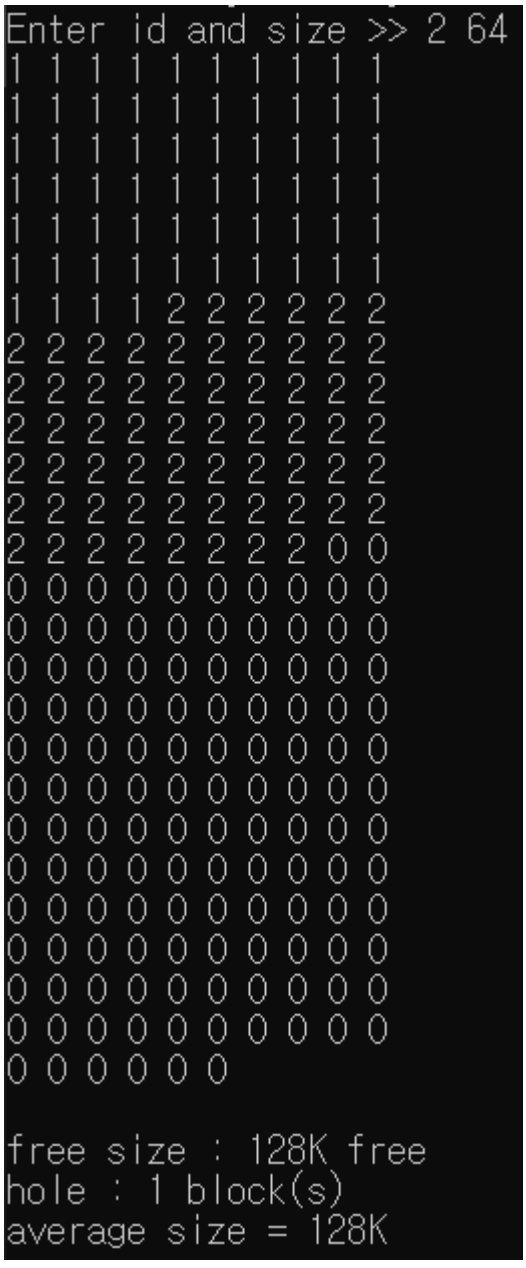
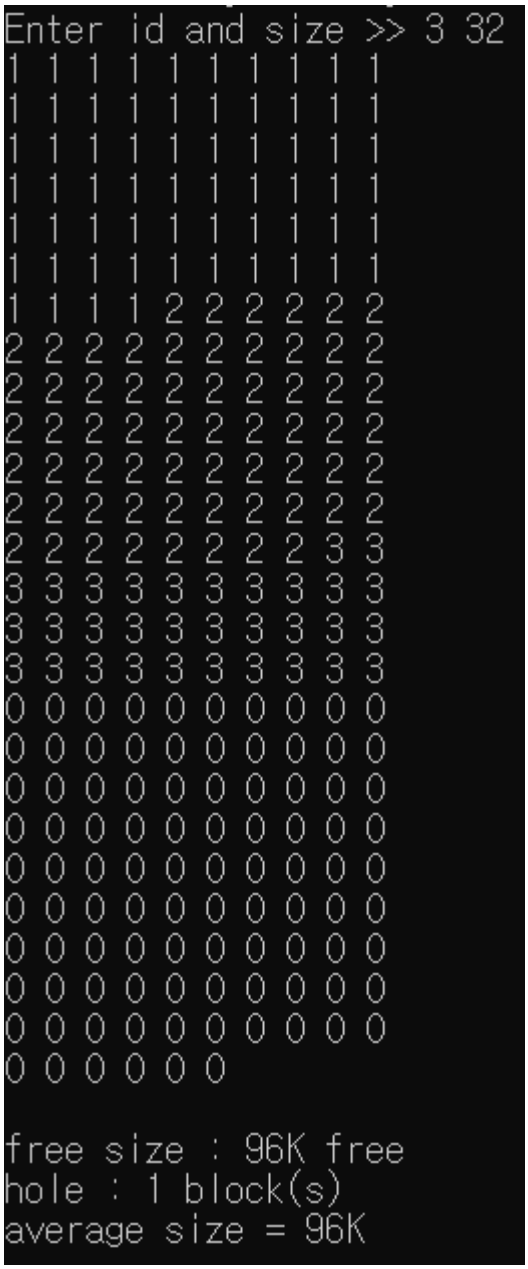
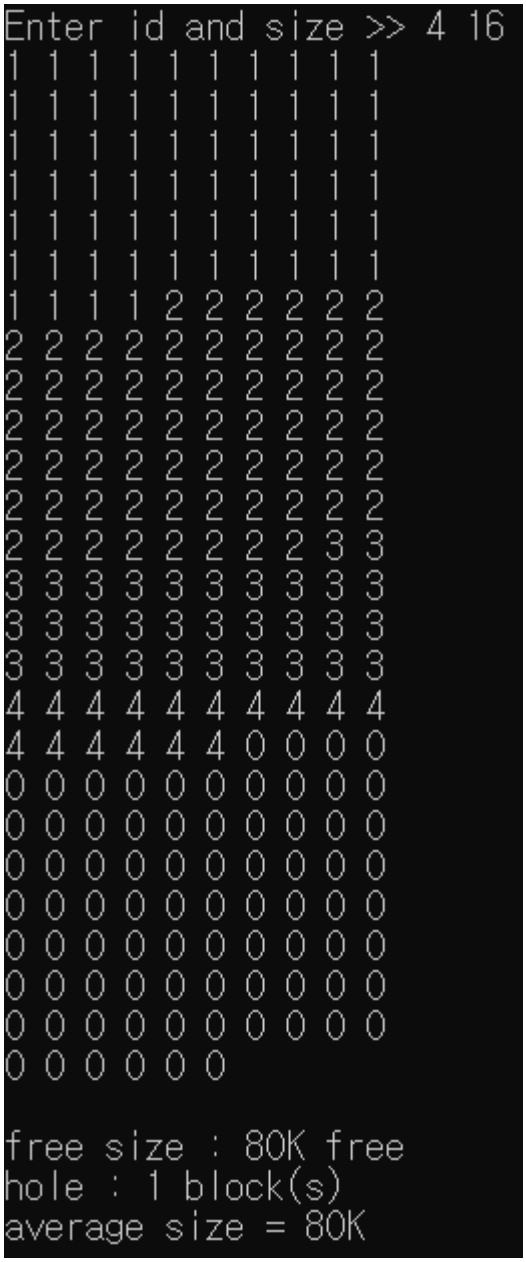
실행결과

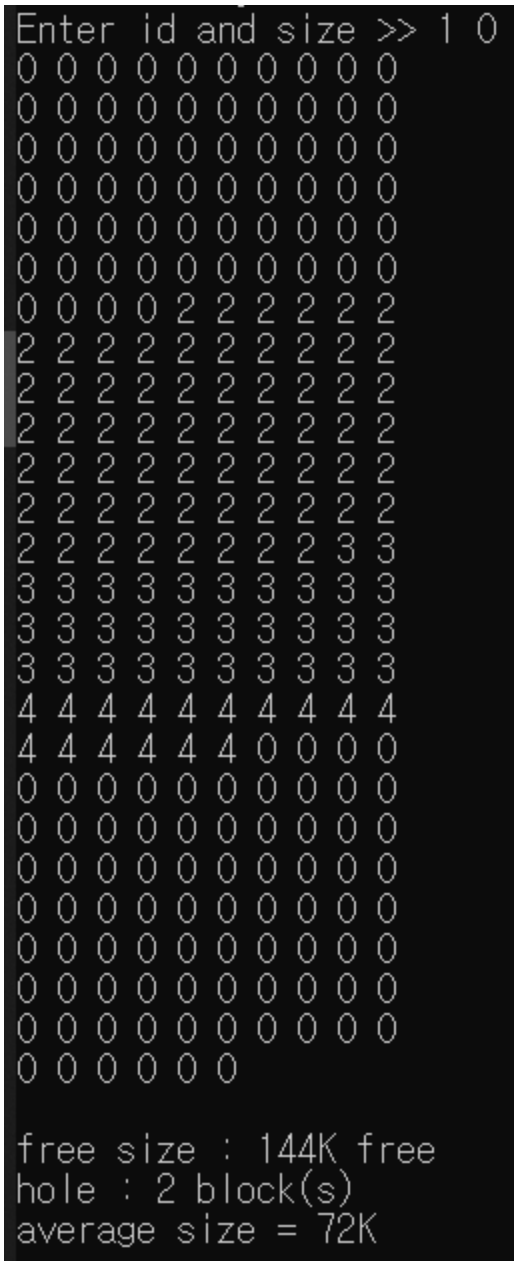

Enter memory size >> 256
Enter id and size >> 1 64
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0
free size : 192K free
hole : 1 block(s)
average size = 192K


Enter id and size >> 2 64
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0
free size : 128K free
hole : 1 block(s)
average size = 128K

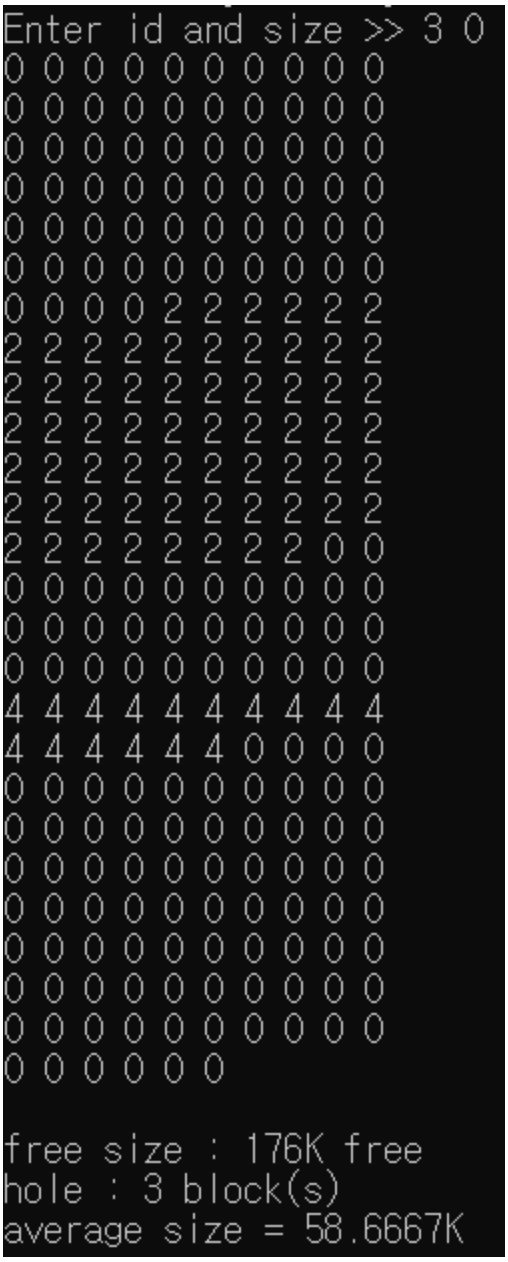




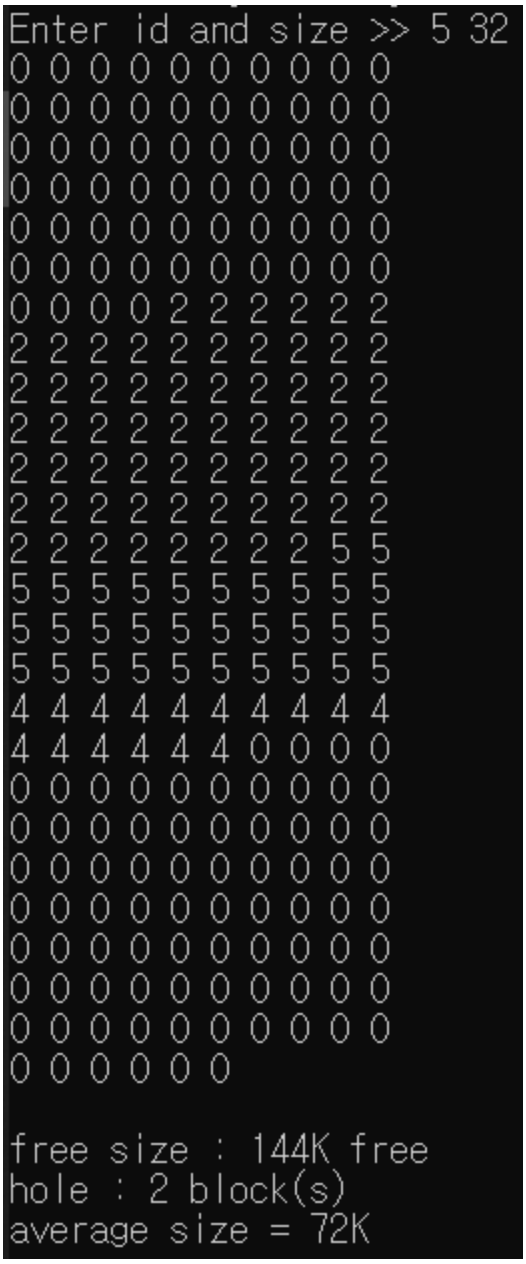
1) release 하기 전까지는 순차적으로 memory 에 process 를 할당한다.
2) 이때 0 은 hole block 을 의미하고 각각의 process 를 allocate 할 때는 memor 에 id 로 채운다.
3) 아직은 release 하지 않았기 때문에 뒷부분의 남은 memory 영역만이 hole block 이다.
   따라서 hole block 의 개수는 1 개이다.
4) free size 는 전체 memory 영역에서 process 가 할당된 영역을 뺀 값이다.
5) average size 는 free size 에서 hole block 의 개수로 나눈 값이다.



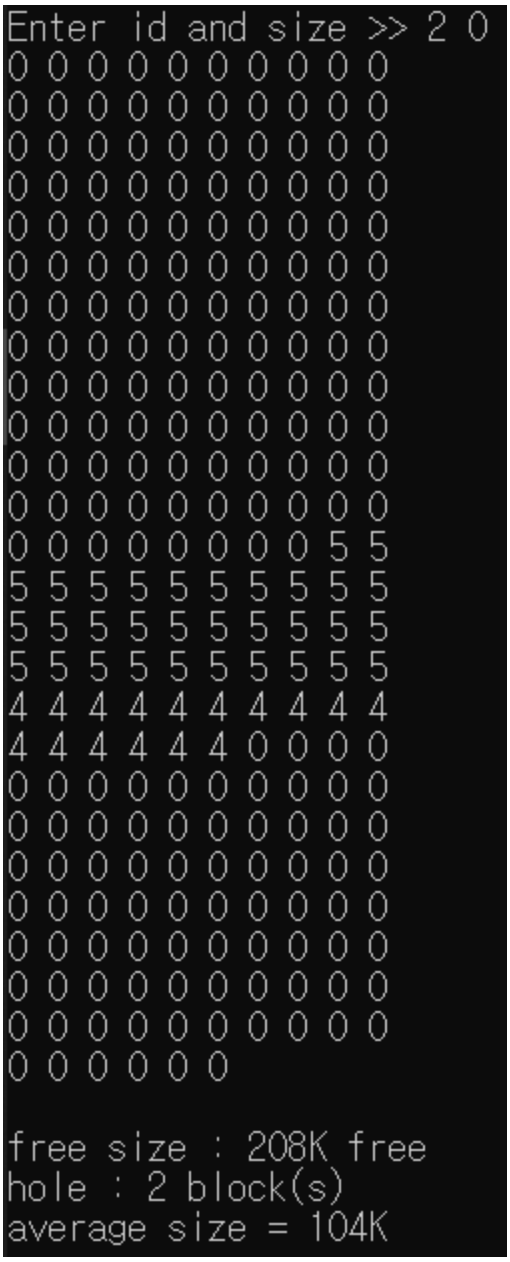
1) process 1 을 release 했기 때문에 원래 process 1 이 있던 부분은 hole 이 된다.
2) hole 의 개수는 process 1 을 release 한 부분과 뒷부분에 남은 memory 영역, 합쳐서 2 개이다.
3) 따라서 전체영역에서 process 가 할당된 영역을 제외한 free size 에서 block size 로 나누면 average size 는 72 이다.



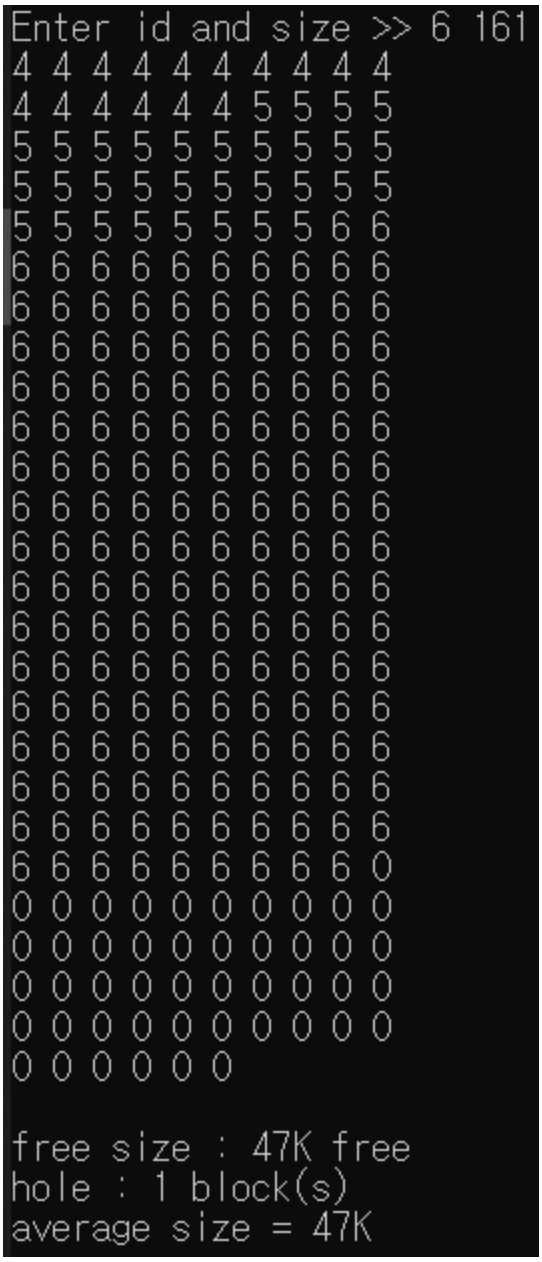
1) process 3 을 release 하면 원래 process 3 가 있던 공간에 hole block 이 생긴다.
2) 따라서 hole block 의 총합은 3 개이다.
3) 전체 memory 영역에서 프로세스가 할당된 영역을 제외한 값에 hole 개수로 나누면 average size 가 계산된다. 이 경우는, {256-(64+16)}/3 = 58.6667 이다.



1) process 5 를 32K 만큼 할당하라는 request 를 받았다.
2) 이때 hole 이 3 개 존재하고, 해당 request 는 best fit 으로 처리되어야 한다.
3) best fit 은 request size 가 들어갈 수 있는 가장 작은 hole 에 할당하는 방법이다.
4) process 5 를 할당하기 전에 hole size 는 위에서부터 64K, 32K, 80K 이므로, 32K 짜리 hole 에 process 5 를 할당한다.
5) 32K 짜리 hole 에 process 5 를 할당하면 process 5 의 size 도 32K 이므로 크기가 딱 들어맞아서 hole 의 개수는 2 개로 줄어든다.



1) process 2 를 release 하라는 명령을 받았다.
2) 이때 process 2 는 process 5 바로 위에 할당되어 있기 때문에 release 해도 hole 이 생기지 않는다.
3) 따라서 hole block 개수는 그대로 2 개이다.



1) process 6 을 161K size 로 할당하라는 명령을 받았다.
2) process 6 을 할당하기 전 hole 의 크기는 위에서부터 128K, 80K 이다.
3) 따라서 이대로는 161K 짜리 process 6 을 할당할 수 없다.
4) compaction 을 수행해야 한다.
5) compaction 을 수행하고 나면 hole 은 1 개가 된다.

1) 만약 조각난 hole 들의 총합보다 request 된 size 가 더 클 경우 해당 process 를 allocate 할 수 없다는 message 를 출력한다. (예외처리)

코드설명

```
cout << "Enter memory size >> "; int size_mem; cin >> size_mem;
vector<int> mem;
for (int i = 0; i < size_mem; i++) mem.push_back(0);

int curr_idx = 0, cnt_hole = 0;
vector<int> hole_size; vector<int> start_idx_of_hole;
vector<int> process_only;
```

1) memory, process, hole 을 관리하기 위해서 stack type 의 vector<int> 을 사용하였다.

```
// process vector release // vector erase
vector<int>::iterator it = find(process_only.begin(), process_only.end(), id);
int release_size = end_release_idx - start_release_idx + 1;
process_only.erase(it, it + release_size);
```

2) release 할 때 release 할 process 의 id 를 찾기 위해서 vector<int> type 의 iterator 를 사용하였다.

```
void sort(vector<int>&_hole_size, vector<int>& _start_idx_of_hole) {
    int tmp;
    for (int i = 0; i < _hole_size.size() - 1; i++) { // bubble sort
        for (int j = i + 1; j < _hole_size.size(); j++) {
            if (_hole_size.at(i) > _hole_size.at(j)) {
                tmp = _hole_size.at(i); _hole_size[i] = _hole_size[j]; _hole_size[j] = tmp; // sort _hole_size
                tmp = _start_idx_of_hole.at(i); _start_idx_of_hole[i] = _start_idx_of_hole[j]; _start_idx_of_hole[j] = tmp; // sort _start_idx_of_hole
            }
        }
    }
}
```

3) best fit 을 수행할 때 bubble sort 를 이용하여 hole size 를 오름차순으로 정렬하였다.