1. Ram.h 헤더파일

```
class Ram {
        char mem[100 * 1024];
        int size;
public:
        Ram();
        ~Ram();
        char read(int address);
        void write(int address, char value);
};
```

2. Ram.cpp C++파일

```
#include <iostream>
#include "Ram.h"
using namespace std;

Ram::Ram() {
	size = 100 * 1024;
	for (int i = 0; i < size; i++)
		mem[i] = 0;
}
Ram::~Ram() {
	cout << "메모리 제거됨" << endl;
}
char Ram::read(int address) {
	return mem[address];
}
void Ram::write(int address, char value) {
	mem[address] = value;
}
```

메모 포함[유김1]: 반복문을 이용해서 0부터 size번째까지의 mem배열 초기화

메모 포함[유김2]: 소멸자에서 "메모리 제거됨" 문자열 출력

메모 포함[유김3]: 메인함수에서의 인자 전달 후 mem 배열의 address번째 값 리턴

메모 포함[유김4]: 메인함수에서의 인자 전달 후 mem 배열의 address번째의 자리에 value값 저장

3. main.cpp 메인함수

```
#include <iostream>
#include "Ram.h"
using namespace std;

int main() {
	Ram ram;
	ram.write(100, 20);
	ram.write(101, 30);
	char res = ram.read(100) + ram.read(101);
	ram.write(102, res);
	cout << "102 번지의 값 = " << (int)ram.read(102) << endl;
}
```

1. 배열 초기화
   for 반복문을 이용해서 0번째부터 size번째까지의 mem배열에 0을 대입해서 초기화

2. 소멸자에서의 문자열 출력
   using namespace std;를 선언하고, cout << "메모리 제거됨" << endl 구문으로 해당 문자열을 출력

3. 메서드에서의 인자 전달 후 배열 접근
   address 주소의 메모리 바이트를 리턴해야하기 때문에,
   address번째의 mem배열에 접근해야함.

   ```
   char Ram::read(int address) {
           return mem[address];
   }
   ```

   int타입의 address값번째의 배열에 접근해서 해당하는 값을 리턴.

4. 메서드에서의 인자 전달 후 배열 접근
   Address주소에 한 바이트로 value를 저장해야함

   ```
   void Ram::write(int address, char value) {
           mem[address] = value;
   }
   ```

   write메서드는 인자를 2개 가지며, int타입 인자와 char타입 인자를 가짐.

   mem배열에서 int타입의 address번째에 있는 위치에 char타입 value를

   저장.

메모 포함[유김5]: 메인함수에서의 인자 전달 후 mem 배열의 address번째 값 리턴

메모 포함[유김6]: 메인함수에서의 인자 전달 후 mem 배열의 address번째의 자리에 value값 저장