## ❒ 개념 확인

(1) 빈 괄호를 채워 넣으세요.

① ( ) 연산자는 변수의 주소를 추출하기위해 사용한다.
② 동적으로 할당된 메모리를 반환하고자 할 때는 ( ) 연산자를 사용한다.
③ ( ) 연산자는 할당되는 동적 메모리의 시작 주소를 반환한다.
④ 동적으로 할당되는 메모리는 ( ) 영역에 할당 받는 메모리이다.
⑤ new를 이용하여 할당 받은 메모리를 해제 할 때에는 ( )를 사용한다.
⑥ 객체 포인터로 멤버를 접근할 때에는 ( ) 연산자를 사용한다.
⑦ ( )은 null pointer를 의미하는 것으로 NULL 매크로 사용시 함수 매개변수로 전달하는 경우 int타입으로 추론되는 문제점을 해결할 수 있다.
⑧ new 연산자를 사용하여 객체를 동적으로 할당할 때 ( )가 호출된다.
⑨ 동적으로 할당 된 객체 소멸 시 ( )가 호출된다.
⑩ 함수 선언 시 ( )를 사용하면 멤버 변수의 값을 변경할 수 없다.
⑪ 스마트 포인터 중 ( )는 포인터를 공유할 수 없다.
⑫ 스마트 포인터를 사용하려면 ( ) 헤더 파일이 필요하다.

(2) 다음 질문에 O, X로 답하세요.

① 동적 메모리 할당을 위해 new 함수를 사용한다. ( )
② 배열은 동적 할당 시 초기화를 할 수 없다. ( )
③ 동적 메모리 반환 순서는 생성 순서와 동일해야 한다. ( )
④ 객체 배열 생성 시 기본 생성자를 호출한다. ( )
⑤ 동적으로 배열을 생성하면서 초기화 할 때 배열 크기는 생략할 수 있다. ( )
⑥ 객체 포인터 변수는 초기화 없이 사용할 수 있다. ( )
⑦ delete 연산자는 정적으로 할당된 메모리를 해제할 때도 사용할 수 있다. ( )
⑧ 배열 형태로 동적 생성한 것은 배열 형태로 삭제해야 한다. ( )
⑨ 클래스 멤버 변수에 대한 동적 생성은 생성자에서 할당하고, 소멸자에는 멤버 변수에 대한 동적 메모리를 해제 한다. ( )
⑩ 스마트 포인터는 할당된 메모리를 자동으로 해제한다. ( )
⑪ 스마트 포인터는 메모리누수와 같은 문제를 해결하기 위해 사용한다. ( )
⑫ this는 전역 함수에서 사용할 수 있다. ( )
⑬ this는 static 멤버 함수에서 사용할 수 없다. ( )
⑭ this는 컴파일러가 삽입해주는 전역변수이다. ( )
⑮ 스마트포인터는 매개변수로 전달할 수 있다. ( )

(3) 제시된 클래스에 대하여 질문에 답하세요

```
class Rec{
        int w, h;
    public:
        int getW();
```

```
    int getH();
    Rec(){   }
    Rec(int a, int b) : w(a), h(b){}
    void write();
};

int main(){
    Rec r(3,4);
}
```

① Rec 클래스에 대한 포인터 변수 p를 선언하세요.

② 선언된 포인터 변수 p에 객체 r의 주소를 지정하세요.

③ 포인터 변수 p를 이용하여 write 함수를 호출할 수 있는 두 가지 방법을 제시하세요.

④ 크기가 4인 Rec 객체 배열 arr를 동적으로 생성하는 문장을 new연산자를 사용하여 제시하세요.

⑤ 크기가 4인 Rec 객체 배열 arr를 동적으로 생성하는 문장을 공유가 허락되지 않는 스마트 포인터를 사용하여 제시하세요.

⑥ 4번에서 new 연산자로 할당 받은 동적메모리를 반환하는 문장을 제시하세요.

⑦ 4번에서 생성된 배열에서, 배열 원소 두번째에 저장된 객체의 write() 멤버 함수를 호출하는 문장을 두 가지 방법으로 제시하세요. 단, 배열 원소 참조 시 []는 사용하지 않습니다.

⑧ 크기가 3인 Rec 객체 배열 dim을 동적으로 할당하면서 매개변수가 있는 생성자를 사용하여 초기화하는 문장을 제시하세요. 단, 초기화 값은 본인이 임의로 결정합니다.

⑨ 다음과 같이 초기화 된 객체 배열을 사용하여 멤버 함수 wirte()를 호출하는 문장을 제시하세요. 단, 범위기반 for를 사용하세요.

```
Rec array[] = { Rec(13,6), Rec(5,8), Rec(3,12) };
```

(4) 다음 프로그램의 실행 결과를 제시하세요.

```
#include <iostream>
using namespace std;
void fun (int* x){
    cout << *(x + 2);
}
int main (){
   int sample[] = {0, 10, 20, 30, 40};
   fun (sample);
   return 0;
}
```

(5) 다음 코드의 실행 결과를 제시하세요.

```
int sample [5] = {5, 10, 15, 20, 25};
cout << *sample + 2 << endl;
cout << *(sample + 2);
```

(6) 스마트 포인터 종류에 대하여 설명하세요.

(7) 아래 설명에 해당하는 배열을 생성하고, 실행 화면처럼 동작하는 프로그램을 작성하세요.

- 사용자로부터 배열의 크기를 입력 받아, 실수를 저장할 수 있는 배열을 동적 할당 받습니다.
- 이후, 배열의 크기만큼 값을 입력 받아, 입력 받은 값으로 배열을 초기화합니다.
- 동적 배열의 생성은 vector를 이용하는 방법과 new를 이용하는 방법 모두 구현합니다.

```
array size? 3
== vector array ==
value? 1.1
value? 1.2
value? 1.3
varr[0]=1.1
varr[1]=1.2
varr[2]=1.3
== new array ==
value? 2.1
value? 2.2
value? 2.3
narr[0]=2.1
narr[1]=2.2
narr[2]=2.3
```

(8) 주어진 메인 함수와 실행 화면을 참고하여 배열을 조금 더 쉽게 사용할 수 있는 Array 클래스를 작성하고, 실행 화면처럼 동작하는 프로그램을 작성하세요.

- Array 클래스는 데이터 멤버로 capacity, size, arr을 갖습니다.
  - capacity(데이터를 담을 수 있는 최대 용량),
  - size(데이터가 저장된 수),
  - arr(힙에 생성한 배열의 첫 번째 요소를 가리키는 포인터)

- 배열 마지막 위치에 요소를 추가하는 멤버 함수 insert()를 갖습니다. capacity를 넘어서 요소를 추가할 경우에는 '배열이 가득 차 요소를 추가할 수 없다'는 메시지를 출력합니다.
- 배열의 요소를 출력하는 멤버 함수 print()도 갖습니다.

```
array size? 4
arr =   10   11   12   13
데이터 34는 추가할 수 없습니다. 배열이 가득 찼습니다
```

```
int main(){
    int count;
    cout << "array size? ";
    cin >> count;

    Array array1(count); //크기가 count 인 배열 생성

    for (int i = 0; i < count; i++) { array1.insert(i+10); }
    array1.print();
    array1.insert(34);
    cout << endl;
    return 0;
}
```

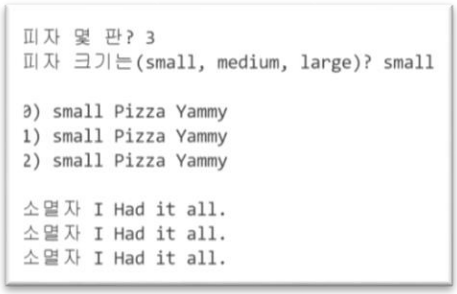
(9) 제시된 메인 함수와 실행 화면을 참고하여 Person 클래스를 작성하세요.

```
p1) name = benny, age = 17
main() : p1.use_count() : 1
p1) name = daniel, age = 17
p2) name = daniel, age = 17
main() : p1.use_count() : 2
메모리 해제
```

```
#include <memory>
#include <iostream>
class Person {
};
int main() {
    auto p1 = std::make_shared<Person>("benny", 17);
    p1->disPlay("p1");
    std::cout << "main() : p1.use_count() : " << p1.use_count() << std::endl;

    std::shared_ptr<Person> p2 = p1;
    p2->changeName("daniel");
    p1->disPlay("p1");
    p2->disPlay("p2");
    std::cout << "main() : p1.use_count() : " << p1.use_count() << std::endl;
}
```

(10) 다음 제시된 클래스를 사용하여 실행 화면과 같이 동작하는 프로그램을 작성하세요. Pizza 객체 배열은 입력 받은 피자 판의 개수만큼 new를 이용하여 동적으로 생성합니다.




```
class Pizza {
    string *size;
public:
    Pizza() = default;
    ~Pizza();
    void setSize(string s); //s 를 size 에 대입
    string getSize();
};
```