Group Activity 06

(3인 혹은 4인으로 팀을 구성하여 아래의 문제를 푼다. 팀 구성은 매 시간마다 달라져도 된다.)

팀원1: _____

팀원2: _____

팀원3: _____

팀원4: _____

1. 다음 프로그램의 출력은? 컴파일 오류나 실행 오류가 나는 경우에는 이유를 간략히 설명하라.

Program	Output

`void fun(int *ptr) {`
`  *ptr = 30;`
`}`
 	
`int main() {`
`  int y = 20;`
`  fun(&y);`
`  cout << y << endl;`
`  return 0;`
`}`
	30

`int main()  {`
`  int *ptr;`
`  int x;`
 	
`  ptr = &x;`
`  *ptr = 0;`
 	
`  cout << "x = " << x << endl;`
`  cout << "*ptr = " << *ptr << endl;`
 	
`  *ptr += 5;`
`  cout << "x = " << x << endl;`
`  cout << "*ptr = " << *ptr << endl;`
 	
`  (*ptr)++;`
`  cout << "x = " << x << endl;`
`  cout << "*ptr = " << *ptr << endl;`
 	
`  return 0;`
`}`	

`int main() {`
`  float arr[5] = {12.5, 10.0, 13.5, 90.5, 0.5};`
`  float *ptr1 = &arr[0];`
`  float *ptr2 = ptr1 + 3;`
 	
`  cout << *ptr2 << endl;`
`  cout << ptr2 - ptr1 << endl;`
 	
`  return 0;`
`}`	

`int main()`
`{`
`  int a;`
`  char *x;`
`  x = (char *) &a;`
`  a = 512;`
`  x[0] = 1;`
`  x[1] = 2;`
`  cout << a << endl;`
`  return 0;`
`}`
	513

`void f(int *p, int *q) {`
`  p = q;`
`  *p = 2;`
`}`
 	
`int i = 0, j = 1;`
`int main() {`
`  f(&i, &j);`
`  cout << i << " " << j;`
`  return 0;`
`}`
	0 2

`int f(int x, int *py, int **ppz) {`
`  int y, z;`
`  **ppz += 1;`
`  z = **ppz;`
`  *py += 2;`
`  y = *py;`
`  x += 3;`
`  return x + y + z;`
`}`
 	
`int main() {`
`  int c, *b, **a;`
`  c = 4;`
`  b = &c;`
`  a = &b;`
`  cout << f(c, b, a));`
`  return 0;`
`}`
	19

`void fun(int *p) {`
`  int q = 10;`
`  p = &q;`
`}`
 	
`int main() {`
`  int r = 20;`
`  int *p = &r;`
`  fun(p);`
`  cout << *p;`
`  return 0;`
`}`
	20

`void fun(int *a) {`
`  a = new int(4);`
`}`
 	
`int main() {`
`  int *p;`
`  fun(p);`
`  *p = 6;`
`  cout << *p;`
`  return 0;`
`}`
	6

`int main() {`
`  int arr[10];`
`  arr[0] = 10;`
`  int *ptr = &arr[0];`
`  arr[1] = 20;`
`  arr[0] = 30;`
`  cout << *ptr << endl;`
`  return 0;`
`}`	

`int main() {`
`  vector<int> vec;`
`  vec.push_back(10);`
`  int *ptr = &vec[0];`
`  vec.push_back(20);`
`  vec[0] = 30;`
`  cout << *ptr << endl;`
`  return 0;`
`}`
	1694433344

`class Foo {`
`public:`
`  int x;`
`  Foo() = default;`
`  Foo(int a): x(a) {}`
`};`
 	
`int main() {`
`  Foo f(10);`
`  Foo *ptr = &f;`
`  cout << *ptr.x << endl;`
`  return 0;`
`}`	

`class Foo {`
`public:`
`  int x;`
`  Foo() = default;`
`  Foo(int a): x(a) {}`
`};`
 	
`int main() {`
`  Foo *f = new Foo(20);`
`  cout << f->x << endl;`
`  return 0;`
`}`	

`class Foo {`
`public:`
`  int x;`
`  Foo() = default;`
`  Foo(int a): x(a) {}`
`};`
 	
`class Bar {`
`public:`
`  Foo *ptr;`
`  int z;`
`  Bar() = default;`
`  Bar(Foo *p, int a): ptr(p), z(a) {}`
`};`
 	
`int main() {`
`  Bar b(new Foo(10), 20);`
`  cout << (*b.ptr).x << endl;`
`  cout << b.ptr->x << endl;`
`  return 0;`
`}`
	10 10

`class Foo {`
`public:`
`  int x;`
`  Foo() = default;`
`  Foo(int a): x(a) {}`
`};`
 	
`int main() {`
`  vector<Foo *> vec {new Foo(1), new Foo(2),`
`  new Foo(3), new Foo(4)};`
`  auto it = vec.begin() + 2;`
`  cout << (*it)->x << endl;`
`  return 0;`
`}`
	3

2. 다음 4개의 함수들 중에서 문제가 있는 것을 모두 고르면? 이유는?

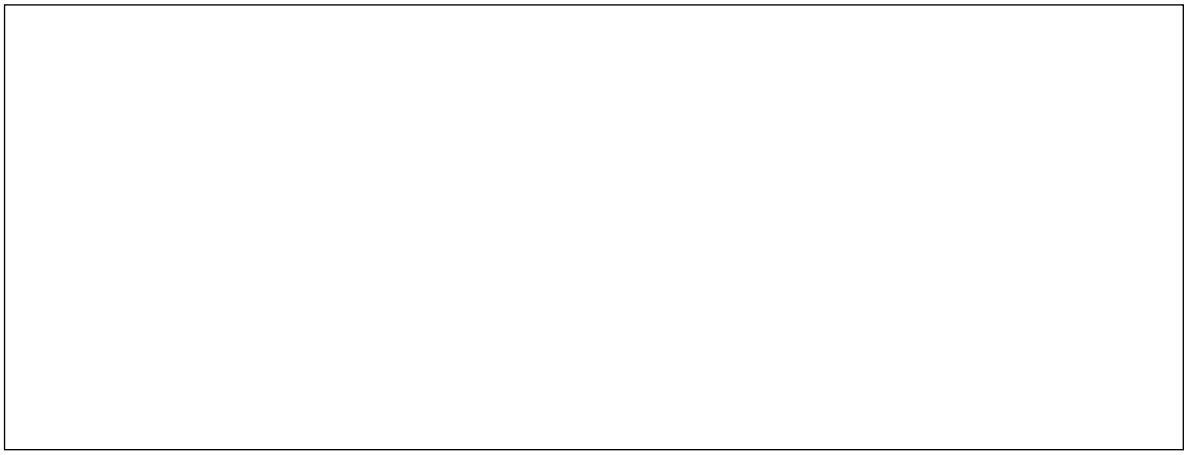
```
int *g() {
    int x = 10;
    return &x;
}

int *g() {
    int *px;
```

```
    *px = 10;
    return px;
}

int g() {
    int *px;
    px = new int;
    *px = 10;
    return *px;
}

int *g() {
    int *px;
    px = new int;
    *px = 10;
    return px;
}
```



3. 동적으로 할당된 배열의 크기를 2배로 늘리는 일을 보통 배열 재할당(array reallocation) 혹은 array doubling이라고 부른다. 다음의 프로그램은 사용자가 `-1`을 입력할 때 까지 사용자로부터 정수를 입력받아 순서대로 배열에 저장한 후 모두 출력하는 일을 한다. `add_to_array` 함수에서는 사용자가 입력한 정수 `k`를 배열 `arr`에 추가한다. 전역변수 `capacity`와 `n`은 각각 현재 배열의 크기와 배열에 저장된 정수의 개수이다. 만약 `capacity==n`면 정수 `k`를 저장하기 위해서 더 큰 크기의 새로운 배열을 동적으로 할당하여야 한다. 예를 들어 `capacity==n==0`이라면 크기가 1인 새로운 배열을 동적으로 할당한 후 포인터 `arr`에 그 주소를 저장한다. `capacity==n!=0`이라면 크기가 `2*capacity`인 새로운 배열을 동적 생성한 후 배열 arr에 저장된 데이터를 새로운 배열로 복사하고, 포인터 `arr`가 새로운 배열을 가리키도록 한다. 이 과정에서 쓰레기(garbage)가 생성되어서는 안된다. 이런 일을 하는 함수 `add_to_array`를 완성하라.

```
const int init_capacity = 1;
int *arr = nullptr;
int capacity = 0;   // size of array arr
int n = 0;          // number of integers stored in arr

int main() {
    int k;
    while(1) {
        cin >> k;
        if (k==-1) break;
        add_to_array(k);
    }
    for (int i=0; i<n; i++)
        cout << arr[i] << " ";
    cout << endl;
    return 0;
}

void add_to_array(int k) {









}
```