1. Explain result

#include <iostream>

using namespace std;

typedef int \*IntPtrType;

int main()

{

IntPtrType ptr\_a, ptr\_b, \*ptr\_c;

ptr\_a = new int; // trỏ ptr\_a tới vùng nhớ ngẫu nhiên còn trống

\*ptr\_a = 3; //ghi giá trị là 3 cho vùng nhớ đó

ptr\_b = ptr\_a; // trỏ ptr\_b tới cùng vùng nhớ mà ptr\_a đang trỏ

cout << \*ptr\_a << " " << \*ptr\_b << "\n"; // \*ptr\_a = \*ptr\_b = 3

ptr\_b = new int; // trỏ ptr\_b tới vùng nhớ mới

\*ptr\_b = 9; // ghi giá trị 9 cho vùng nhớ đó

cout << \*ptr\_a << " " << \*ptr\_b << "\n"; // \*ptr\_a = 3, \*ptr\_b = 9

\*ptr\_b = \*ptr\_a; // gán giá trị của vùng nhớ mà ptr\_b đang trỏ cho vùng nhớ mà

//ptr\_a đang trỏ

cout << \*ptr\_a << " " << \*ptr\_b << "\n"; // \*ptr\_a = \*ptr\_b = 3;

delete ptr\_a; // thu hồi quyền quản lý của con trỏ ptr\_a với vùng nhớ lúc này con

// con trỏ a đang bơ vơ không có nới trỏ tới.

ptr\_a = ptr\_b; // trỏ ptr\_a tới vùng nhớ mà ptr\_b đang trỏ

cout << \*ptr\_a << " " << \*ptr\_b << "\n"; //\*ptr\_a = \*ptr\_b = 3

ptr\_c = &ptr\_a; // trỏ p-to-p ptr\_c tới ptr\_a

cout << \*ptr\_c << " " << \*\*ptr\_c << "\n";// \*ptr\_c = ptr\_a = địa chỉ vùng nhớ mà //ptr\_a đang quản lý.

//\*\*ptr\_c = \*ptr\_a = 3 là giá trị tại vùng

// nhớ

delete ptr\_a; // thu hồi quyền quản lý của con trỏ ptr\_a với vùng nhớ

ptr\_a = NULL;// trỏ ptr\_a về NULL

system("pause");

return 0;

}

2.

#include​ ​<iostream>  
using​ ​namespace​ std;  
void​ main()  
{  
int​ a[4] = {1, 2, 3, 4};

int​ \*p = a;

int​ \*p2 = ​new​ ​int​;

*delete​ p;*

*delete​ a;*

mảng a là khai báo tĩnh đang lưu ở stack nên con trỏ p trỏ tới a cũng lưu ở

stack nên nên chương trình sẽ tự thu hồi vùng nhớ khi kết thúc, lập trình viên không can thiệp được, nên không thể delete p và a. ta bỏ 2 mã lệnh này.

delete​ p2;

biến p2 đang lưu ở heap vì là con trỏ động, nên sau khi delete ta phải trỏ p2 về NULL, ta thêm mã lênh

p2 = NULL;

}

3.1 why

Vùng nhớ được cấp phát trên Heap sẽ không tự động hủy bởi chương trình khi kết thúc khối lệnh, việc thu hồi vùng nhớ đã cấp phát trên Heap được giao cho lập trình viên tự quản lý. Nếu trong chương trình có yêu cầu cấp phát bộ nhớ trên Heap mà không được thu hồi hợp lý sẽ gây tràn bộ nhớ hoặc lãng phí.

3.2 when

Khi sử dụng xong vùng nhớ thường thì đó là vùng nhớ sử dụng tạm thời thì sau đó ta sẽ thu hồi ngay.

3.3 Delete dùng để giải phóng con trỏ động quản lý một vùng nhớ, delete[] để giải phóng con trỏ quản lý 1 mảng cấp phát động.

Demo: **int** \*a = **new** **int**;

**delete** a;

**char** \*arr = **new** **char**[10];

**delete**[] arr;

4.

1. \*p1 == 10  
2. \*p2 =0x100 = p1 = &iVar  
3. \*(\*p2) = \*p1 = iVar = 10