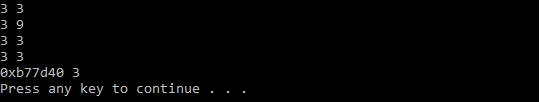
**Câu 1:**

Kết quả của chương trình là:



**Giải thích:**

**ptr\_a = new int;**

**\*ptr\_a = 3;**

**ptr\_b = ptr\_a;**

**cout << \*ptr\_a << " " << \*ptr\_b << "\n";** //Khởi tạo một vùng nhớ trên heap và con trỏ prt\_a trỏ tới, nó lưu vào vùng nhớ này giá trị 3. Ptr\_b = ptr\_a, dòng lệnh này có nghĩa là con trỏ ptr\_b cũng trỏ tới địa chỉ mà con trỏ ptr\_a đang trỏ. Nên khi in ra giá trị của địa chỉ vùng nhớ con trỏ ptr\_a và ptr\_b đang trỏ tới thì đều in ra kết quả là 3.

-------------------------------------------------------------------------------------------------------------------

**ptr\_b = new int;**

**\*ptr\_b = 9;**

**cout << \*ptr\_a << " " << \*ptr\_b << "\n";** //Dòng lệnh ptr\_b = new int, có nghĩa là khởi tạo một vùng nhớ mới trên heap và con trỏ ptr\_b trỏ tới. \*ptr\_b = 9, nghĩa là con trỏ ptr\_b sẽ lưu giá trị 9 vào vùng nhớ nó đang trỏ tới. Nên khi in ra giá trị bên trong vùng nhớ 2 con trỏ đang trỏ tới thì con trỏ a vẫn đang trỏ tới vùng nhớ chứa giá trị 3, còn con trỏ b bây giờ đã trỏ tới vùng nhớ chứa giá trị 9. Nên kết quả là 3 9.

-------------------------------------------------------------------------------------------------------------------

**\*ptr\_b = \*ptr\_a;**

**cout << \*ptr\_a << " " << \*ptr\_b << "\n";** //Dòng lệnh \*ptr\_b = \*ptr\_a, có nghĩa là bây giờ con trỏ b sẽ trỏ tới vùng nhớ mà con trỏ a đang trỏ. Vì a đang trỏ tới vùng nhớ chứa giá trị 3, nên kết quả in ra là 3 3.

-------------------------------------------------------------------------------------------------------------------

**delete ptr\_a;**

**ptr\_a = ptr\_b;**

**cout << \*ptr\_a << " " << \*ptr\_b << "\n";** //Dòng lệnh delete ptr\_a,sẽ có nghĩa là giải phóng vùng nhớ con trỏ a đang trỏ tới và cắt mối liên kết của con trỏ a tới vùng nhớ đó. Nhưng nó vẫn còn giữ địa chỉ của vùng nhớ đã được cấp phát, con trỏ prt\_b đang trỏ tới vùng nhớ chứa giá trị 3 đó thông qua con trỏ prt\_a ở câu lệnh trước đó, nên khi dòng lệnh ptr\_a = ptr\_b, con trỏ prt\_a sẽ trỏ tới lại vùng nhớ nó vừa giải phóng thông qua con trỏ prt\_b. Nên khi in ra kết quả vẫn là 3 3.

-------------------------------------------------------------------------------------------------------------------

**ptr\_c = &ptr\_a;**

**cout << \*ptr\_c << " " << \*\*ptr\_c << "\n";**// Dòng lệnh ptr\_c = &ptr\_a, có nghĩa là con trỏ ptr\_c sẽ lưu địa chỉ của con trỏ ptr\_a tại vùng nhớ nó đang trỏ tới. \*ptr\_c, sẽ in ra địa chỉ của con trỏ ptr\_a đang trỏ tới, còn \*\*ptr\_c sẽ in ra giá trị tại địa chỉ đó.

**Câu 2:**

**Vấn đề:**

Mảng một chiều a và con trỏ p trỏ tới mảng một chiều a đều được cấp phát bộ nhớ tĩnh, nên nó không được dùng từ khóa delete để thu hồi vùng nhớ mà hệ điều hành sẽ tự động làm điều đó. Và sau khi delete nên cho các đối tượng về NULL để nó không trỏ tới các vùng nhớ khác trong bộ nhớ,có thể gây ra lỗi.

**Cách giải quyết:**

Cách 1: Bỏ 2 dòng lệnh delete p và delete a; thêm dòng lệnh p2 = NULL.

Cách 2: Khai báo a là một con trỏ, trỏ tới một mảng được new trên bộ nhớ HEAP (int \*a = new int[4]). Bỏ dòng lệnh delete p, để nguyên dòng lệnh delete a hoặc đổi thành delete []a. Thêm 2 dòng lệnh a = NULL và p2 = NULL.

**Câu 3:**

**Why?**

Bởi vì khi không muốn tiếp tục sử dụng vùng nhớ đã được cấp phát trên HEAP. Ta cần sử dụng từ khóa delete để giải phóng vùng nhớ đó và trả lại cho hệ điều hành để nó có thể cấp phát lại vùng nhớ đó cho đối tượng khác.

**When?**

Chúng ta dùng delete khi đối tượng đó lúc khởi tạo được new ở HEAP và sau đó đối tượng không cần phải làm việc ở vùng nhớ đó nữa.

**Difference?**

Sự khác nhau giữa delete và delete[] là: với việc new ra một đối tượng ở HEAP, thì ta chỉ cần gọi 1 lần hàm hủy để hủy đối tượng đó, nhưng với đối tượng là một mảng, ta cần phải truy cứu kích thước của mảng đó để tìm n phần tử có trong mảng đó và gọi n hàm hủy để hủy đối tượng đó.

**Chương trình demo:**

#include <iostream>

Using namespace std;

int main()

{

int \*a = new int[4];

int \*p = new int;

delete []a;

delete p;

a = NULL;

p = NULL;

system(“pause”);

}

**Câu 4:**

1. \*p1 = 10;
2. \*p2 = 0x200;
3. \*(\*p2) = 10;