Đáp án:

1. A
2. C
3. B
4. C
5. A
6. B
7. D
8. D
9. B
10. D
11. B
12. D
13. B
14. Kết quả in ra: **12**.

Giải thích:

* Dòng 1: Khai báo 3 biến số nguyên (kiểu int) a, b, c bằng cấp phát tĩnh và gán lần lượt các giá trị 1, 2, 3;
* Dòng 2: Khai báo một mảng arr chứ 3 phần tử, mỗi phần tử là một con trỏ kiểu int. 3 phần tử này lần lượt được gán giá trị là địa chỉ của 3 biến a, b, c (tức 3 con trỏ trong mảng lần lượt trỏ tới biến a, b, c);
* Dòng 3: in ra giá trị bằng lệnh cout (cout << \*arr[(\*arr[2])-- - 3];):
  + (\*arr[2])--: truy cập vào giá trị mà con trỏ tại arr[2] đang trỏ tới, tức đang truy cập vào giá trị của biến c, sau đó toán tử postifx -- làm giảm c đi một đơn vị, vậy (\*arr[2])-- ~~=~~ 3 và c = 2;
  + \*arr[(\*arr[2])-- - 3]: lúc này tương đương với \*arr[3 - 3] hay \*arr[0], nghĩa là đang truy cập vào giá trị mà con trỏ arr[0] đang trỏ tới – là giá trị của biến a, a = 1;
  + Tóm lại kết thúc dòng 3:
    - a = 1, b = 2, c = 2;
    - Kết quả in ra: 1;
* Dòng 4: in ra giá trị của biến c bằng lệnh cout, c lúc này bằng 2. Kết thúc dòng 4, kết quả in ra: 2;

1. Đáp án:

* Dòng 1 (“delete p;”):
  + q[2] = <undefined>;
  + p[1][2] = <undefined>;
* Dòng 2 (“delete q;”):
  + q[2] = <undefined>;
  + p[1][2] = <undefined>;
* Dòng 3 (“delete[] q;”):
  + q[2] = <undefined>;
  + p[1][2] = <undefined>;
* Giải thích: việc sử dụng **delete** cùng với **new[]** sẽ gây ra **undefined behavior**, và việc cố xem xét giá trị cần tìm khi đối mặt với *hành động không xác định* đó là không cần thiết.
* Kết quả có thể phụ thuộc vào trình biên dịch và IDE đang sử dụng, chúng ta không thể đoán trước được điều gì xảy ra và cũng như không thể áp đặt bất kỳ lý giải nào trong việc giới hạn đáp án.