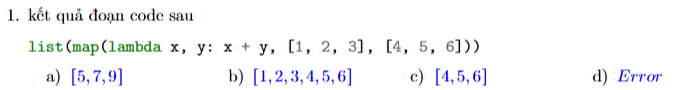
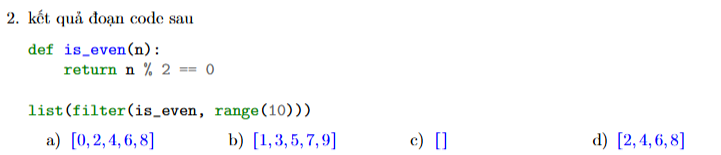
**Câu 1:**



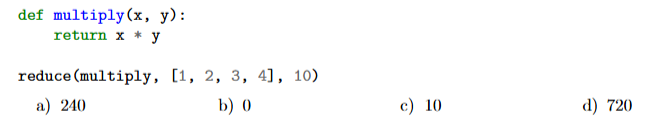
1A

**Câu 2:**



2A. Lời gọi hàm range(10) trả về [0..9], hàm is\_even trả về true nếu n là số chẵn => kết quả filter là list số chẵn từ 0-9.

**Câu 3:**



3A. reduce gồm 3 tham số reduce(function, iterable, initial) trong đó initial là giá trị khởi đầu và nó là tùy chọn(nếu có thì lấy initial và iterable[0] chạy trước, nếu không thì iterable[0] và iterable[1]).

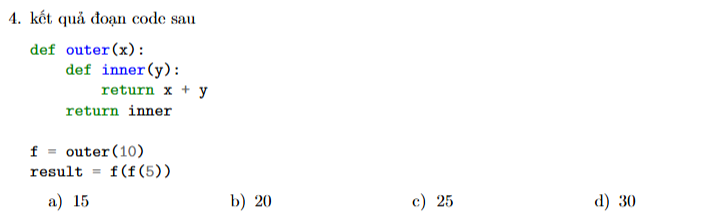
10 và 1 => 10\*1

10\*1 và 2 => 10\*1\*2

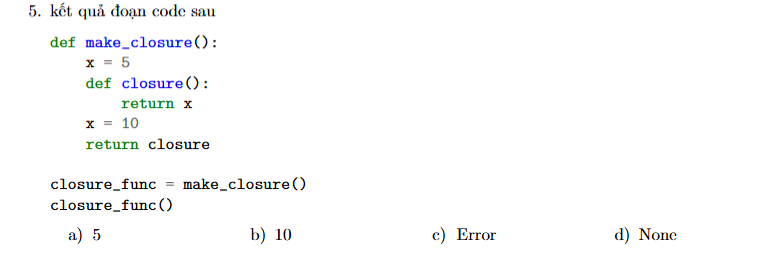
10\*1\*2 và 3 => 10\*1\*2\*3

10\*1\*2\*3 và 4 => 10\*1\*2\*3\*4 = 240. Chọn A

Câu 4:



4C. f = outer(10) = def inner(y): return 10 + y => **f(5) = 15** =>  **f(f(5)) = 25**

****

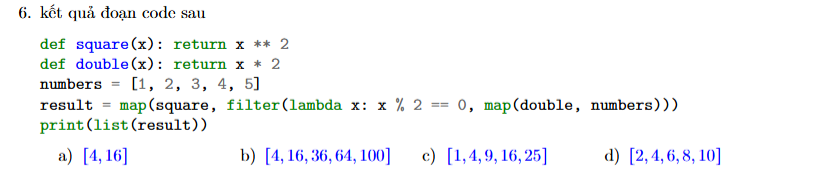
5B. Trong Python, closure giữ tham chiếu đến biến x chứ không phải giá trị của biến tại thời điểm định nghĩa.

Luồng như sau:

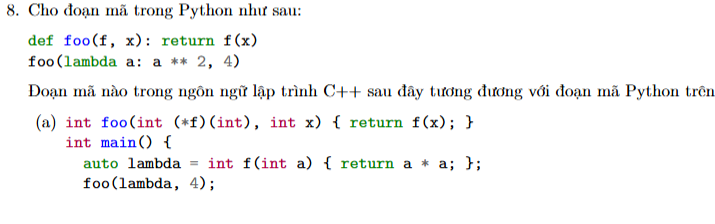
* Lưu obj make\_closure() (lưu lại chứ không thực thi mấy dòng bên trong hàm - late binding trong python)
* Nhảy đến phép gán.
* Lúc này mới thực thi hết các dòng trong hàm make\_closure(). Biến x được thay đổi giá trị thành 10. Hàm trả về closure() giữ tham khảo đến biến x.
* Hàm trả về này đc gọi (dòng cuối) và lấy value của x là 10 => Chọn B.

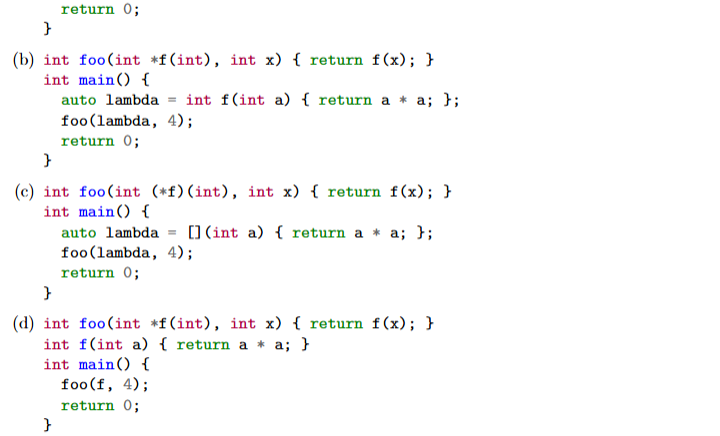
*<visualize on pythontutor>*

Câu 6:



6B. Double toàn numbers => lọc ra số chẵn(100%) => Bình phương mỗi phần tử => Chọn B

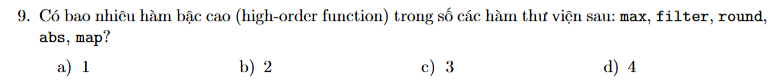
**Câu 8:** 



8C. Cú pháp con trỏ hàm: type (\*fp) ( para\_type\_1, para\_type\_2, para\_type\_3,...);

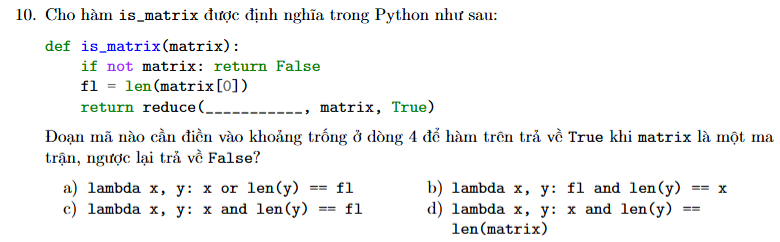
=>kiểu hàm f là **int (\*f)(int)** => loại B và D.

[](int a) { return a \* a; } là cú pháp lambda function đúng trong c++ => Chọn C



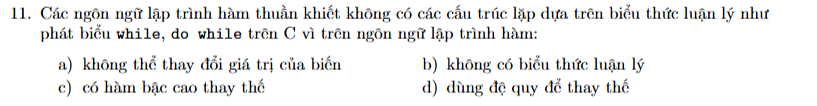
9B. filter,map

**Câu 10:**

****

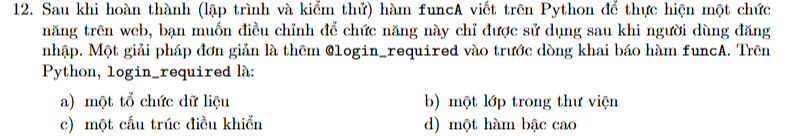
10C. Câu này bản chất là cố sử dụng reduce để kiểm tra tất cả các hàng/cột của matrix bằng nhau(bằng giá trị fl). Tất cả đều True => dùng ‘and’. Xét các tham số hàm lamda x là accumulate, y là current => y là hàng/cột cần check độ dài => Chọn C.

**Câu 11:**

****

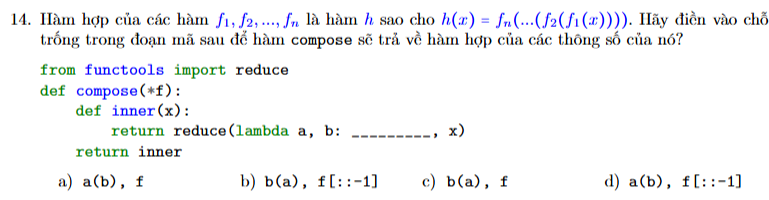
11C. lấy haskell làm ví dụ đi.

**Câu 12:**



12D. là một hàm bậc cao nhận hàm đứng sau nó làm tham số.

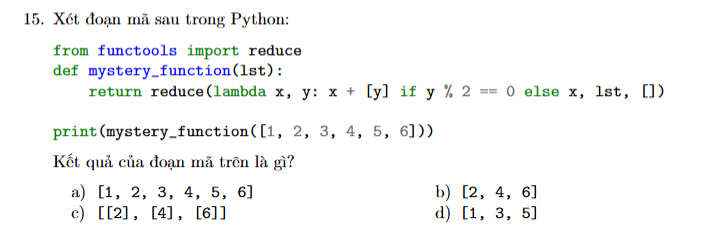
**Câu 14:**



14C.

* Ví dụ: a = compose(f1,f2,f3) =>a(x) là f3(f2(f1(x))) <ngược chiều>
* **\*f** => hàm này nhận vào 1 số lượng tham số kxđ và được gói vào 1 tuple duy nhất f.
* lambda a,b : a là accumulate hiện tại b là phần từ tiếp theo mà phần tử tiếp theo thì phải nhận thằng accumulate phía trước làm tham số => b(a)
* Để đúng thứ tự input phải ngược chiều output thì là: **b(a),f**

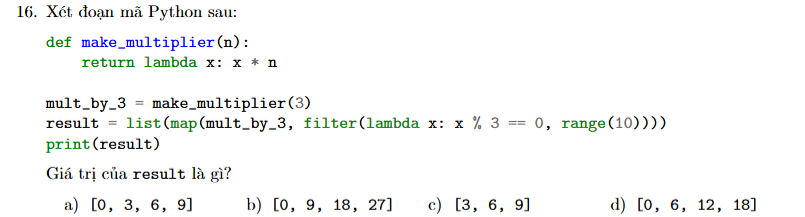
**Câu 15:**



15B. Hàm thêm các phần tử số chẵn vào dãy.

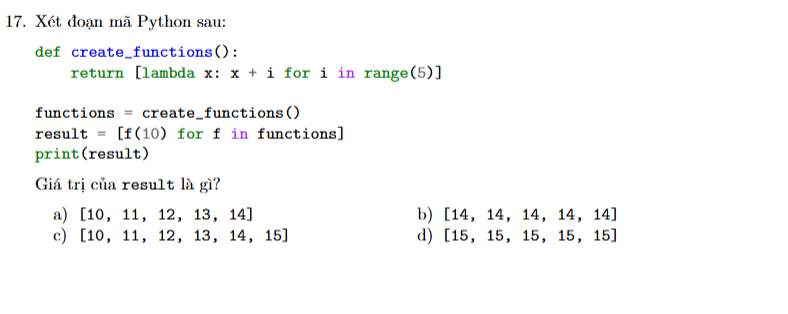
Chú ý nhập nhằng: **x + [y] if y % 2 == 0 else x** nghĩa là **(x + [y]) if y % 2 == 0 else x**

**Câu 16:**

****

16B.

**Câu 17:**

****

17 A.

return [lambda x: x + i for i in range(5)] => trả về list các hàm lambda:

x -> x + 0

x -> x + 1  
..

x -> x + 4

[f(10) for f in functions] => lấy ra từng hàm lamda ở trên rồi truyền tham số 10 cho nó

=> [10+0, 10+1,...,10+4] => Chọn A