Erlang and message passing

What does `f(5)` return?

```
f(0) -> 0;
f(N) -> N + f(N-1).
```

1. 0
2. 5
3. 15
4. the factorial of 5

What does `f(5)` return?

```
f(0) -> 0;
f(N) -> N + f(N-1).
```

1. 0
2. 5
3. 15
4. the factorial of 5

What does `g([a,b,c,d,e,f,g])` return?

```
g([])         -> [];
g([X])        -> [X];
g([X|[Y|T]]) -> [X|g(T)].
```

1. `[]`
2. `[a]`
3. `[a,b,c,d,e,f,g]`
4. `[a,c,e,g]`

What does `g([a,b,c,d,e,f,g])` return?

```
g([])        -> [];
g([X])       -> [X];
g([X|[Y|T]]) -> [X|g(T)].
```

1. `[]`
2. `[a]`
3. `[a,b,c,d,e,f,g]`
4. `[a,c,e,g]`

What do `h({3,3})` and `h({4,3})` return?

```
h({3,B}) -> B;
h({_,3}) -> 3;
h({_,_}) -> 4.
```

1. 3 and 3
2. 3 and 4
3. 4 and 3
4. 4 and 4

What do `h({3,3})` and `h({4,3})` return?

```
h({3,B}) -> B;
h({_,3}) -> 3;
h({_,_}) -> 4.
```

1. 3 and 3
2. 3 and 4
3. 4 and 3
4. 4 and 4

What does `k([])` return?

```
k({_,_,_}) -> [3,3,3];
k(X) ->
  case X of
    {A,B} -> A + B;
    _     -> 0
  end.
```

1. 0
2. `[3,3,3]`
3. It throws an exception
4. `{0,0}`

What does `k([])` return?

```
k({_,_,_}) -> [3,3,3];
k(X) ->
  case X of
    {A,B} -> A + B;
    _     -> 0
  end.
```

1. 0
2. `[3,3,3]`
3. It throws an exception
4. `{0,0}`

What does process Q print?

process P

```
p() ->  % Q is Q’s pid
  Q ! {self(), 0},
  Q ! {self(), 2}.
```

process Q

```
q() ->  % P is P’s pid
  receive {P, N} ->
    io:format("~p", [N+1]) end,
  q().
```

1. 0 and 2, in any order
2. 0 and then 2
3. 1 and then 3
4. 1 and 3, in any order

What does process Q print?

process P

```
p() ->  % Q is Q’s pid
  Q ! {self(), 0},
  Q ! {self(), 2}.
```

process Q

```
q() ->  % P is P’s pid
  receive {P, N} ->
    io:format("~p", [N+1]) end,
  q().
```

1. 0 and 2, in any order
2. 0 and then 2
3. 1 and then 3
4. 1 and 3, in any order

What do processes `P` and `Q` print?

process `P`

```
p() ->  % Q is Q’s pid
  Q ! 0,
  receive {P, N} ->
    io:format("~p", [N+1])
  end.
```

process `Q`

```
q() ->  % P is P’s pid
  P ! 2,
  receive {Q, N} ->
    io:format("~p", [N+1])
  end.
```

1. 0 and 2, in any order
2. 0 and then 2
3. 1 and then 3
4. 1 and 3, in any order

What do processes `P` and `Q` print?

process `P`

```
p() ->  % Q is Q's pid
  Q ! 0,
  receive {P, N} ->
    io:format("~p", [N+1])
  end.
```

process `Q`

```
q() ->  % P is P's pid
  P ! 2,
  receive {Q, N} ->
    io:format("~p", [N+1])
  end.
```

1. 0 and 2, in any order
2. 0 and then 2
3. 1 and then 3
4. 1 and 3, in any order

What does process Q print?

process P

```
p() ->  % Q is Q’s pid
  self() ! self(),
  receive self() ->
    Q !
    {self(),
     fun (Y) -> Y+1 end}
  end.
```

process Q

```
q() ->  % P is P’s pid
  receive {P, F} ->
    io:format("~p", [F(3)]) end.
```

1. 3
2. 4
3. P’s pid (process identifier)
4. Q’s pid (process identifier)

What does process Q print?

process P

```
p() ->  % Q is Q’s pid
  self() ! self(),
  receive self() ->
    Q !
    {self(),
     fun (Y) -> Y+1 end}
  end.
```

process Q

```
q() ->  % P is P’s pid
  receive {P, F} ->
    io:format("~p", [F(3)]) end.
```

1. 3
2. 4
3. P’s pid (process identifier)
4. Q’s pid (process identifier)