

Tên học phần: Nhập Môn CNTT1 **Mã HP:** _____

Thời gian làm bài: 4 tuần **Ngày thi:** _____

Họ tên - MSSV:

Phan Đặng Hoài Bảo - 1712288

Cao Nhơn Hưng - 1712475

Lê Trần Hữu Đắc - 1712026

Điện thoại: 01683544866

Email: caberrrr@gmail.com

Bản tự đánh giá		
Câu	Mức độ hoàn thành (%)	Ghi chú
1	100	
2	100	
3	100	
4	100	
5	100	
6	100	
7	100	
8	100	
9	100	
10	100	
11	100	
12	100	
13	100	
14	100	
15	100	
16	100	
17	100	
18	100	
19	100	
20	100	
21	100	
22	100	

23	100	
24	100	
25	100	
26	100	
27	100	
28	100	
29	100	
30	100	
31	100	
32	100	Sửa đề: tổng chạy từ $i=1$ đến $(n-1)$
33	100	
34	100	
35	100	
36	100	
37	100	
38	100	
39	100	
40	100	
41	100	
42	100	
43	100	
44	100	
45	100	
46	100	
47	100	
48	100	
49	100	
50	30	
51	30	

Câu 1.

- Code:

```
cau01.erl x cau02.erl x cau05.erl x cau07.erl x hel
1 -module(cau01).
2 -export([tamgiac/3]).
3 tamgiac(A,B,C) ->
4 if
5     A + B <= C; B + C <= A; A + C <= B ->
6     io:fwrite("Khong phai tam giac\n")
7 ;A == B, B == C ->
8     io:fwrite("Tam giac deu\n")
9 ;A == B; B == C; C == A ->
10    io:fwrite("Tam giac can\n")
11 ;A*A + B*B == C*C, A*A == B*B; B*B + C*C == A*A, B*B == C*C; C*C + A*A == B*B, C*C == A*A ->
12    io:fwrite("Tam giac vuong can\n")
13 ;A*A + B*B == C*C; B*B + C*C == A*A; C*C + A*A == B*B ->
14    io:fwrite("Tam giac vuong\n")
15 ;true ->
16    io:fwrite("Tam giac thuong\n")
17 end.
```

- Giải thích:

- Hàm tamgiac kiểm tra các điều kiện có phải tam giác hay không, nếu phải thì là tam giác gì và in ra màn hình.

- Kết quả:

```
Erlang
File Edit Options View Help
Erlang/OTP 20 [erts-9.1] [64-bit] [smp:4:4] [ds:4:4:10] [async-threads:10]
Eshell V9.1 (abort with ^G)
1> cd("F:\Erlang Projects").
F:/Erlang Projects
ok
2> c(cau01).
{ok,cau01}
3> cau01:tamgiac(1,2,5).
Khong phai tam giac
ok
4> cau01:tamgiac(3,4,5).
Tam giac vuong
ok
5> cau01:tamgiac(6,6,9).
Tam giac can
ok
... █
```

Câu 2.

- Code:

```

1 -module(cau02).
2 -export([vttuongdoi/3]).
3 vttuongdoi(Rn,Rl,D) ->
4 if
5     D > Rn+Rl ->
6         io:fwrite("Tach roi\n")
7     ;D == Rn+Rl ->
8         io:fwrite("Tiep xuc ngoai\n")
9     ;Rn == Rl, D == 0 ->
10        io:fwrite("Chong khop len nhau\n")
11    ;Rl-Rn == D ->
12        io:fwrite("Tiep xuc trong\n")
13    ;D+Rn < Rl ->
14        io:fwrite("Bao nhau\n")
15    ;Rn+Rl > D ->
16        io:fwrite("Giao nhau\n")
17 end.

```

- Giải thích:

- Hàm vttuongdoi kiểm tra các vị trí tương đối của 2 đường tròn và in ra màn hình.

- Kết quả:

```

Erlang
File Edit Options View Help
Tam giac can
ok
6> c(cau02).
{ok,cau02}
7> cau02:vttuongdoi(5,6,15).
Tach roi
ok
8> cau02:vttuongdoi(5,6,11).
Tiep xuc ngoai
ok
9> cau02:vttuongdoi(5,6,0).
Bao nhau
ok

```

Câu 3.

- Code:

```

1 -module(cau03).
2 -export([areaC/1,areaT/1,area/2]).
3
4 areaC(R) -> R*R*3.14.
5
6 areaT(A) ->
7     math:sqrt(1.5*A*0.5*A*0.5*A*0.5*A).
8
9 area(A,R) -> 0.5*areaC(R) + 3*areaT(A).
10
11

```

- Giải thích:

- Hàm areaC(R) tính diện tích hình tròn.
- Hàm areaT(A) tính diện tích tam giác đều.

- Diện tích hình cần tính = $0,5 \cdot \text{diện tích hình tròn} + \text{diện tích 3 tam giác đều}$, được thực hiện bởi hàm `area(A,R)`.

- Kết quả:

```

Erlang/OTP 20 [erts-9.1] [64-bit] [smp:8:8] [ds:8:8:10] [async-threads:10]

Eshell V9.1 (abort with ^G)
1> cd("/Erlang_project").
E:/Erlang_project
ok
2> c(cau03).
{ok,cau03}
3> cau03:area(2,4).
30.316152422706633
4> cau03:area(3,6).
68.21134295108993
5> cau03:area(4,5).
60.03460969082653
6>

```

Câu 4.

- Code:

```

1 -module(cau04).
2 -export([dao/1, dao_list/1, gtri/1, doixung/1]).
3
4 dao(0) -> [];
5 dao(N) -> [N rem 10 | dao(N div 10)].
6
7 dao_list([]) -> [];
8 dao_list([H|T]) -> dao_list(T) ++ [H].
9
10 %tính giá trị từ phải qua trái
11 gtri([]) -> 0;
12 gtri([H|T]) -> gtri(T) * 10 + H.
13
14 doixung(N) ->
15     M = gtri(dao_list(dao(N))),
16     if (N == M) -> io:fwrite("~p doi xung~n", [N]);
17         true -> io:fwrite("~p ko doi xung~n", [N])
18     end.

```

- Giải thích:

- Hàm `dao(N)` trả về list chứa các chữ số của N sau khi đảo ngược.
- Hàm `dao_list(L)` trả về list L sau khi đảo ngược.
- Hàm `gtri(L)` trả về giá trị thập phân của list L đảo ngược.
- Hàm `doixung(N)` cho biết N có đối xứng không.

- Kết quả:


```
{ok,cau05}
3> cau05:lietkedoixung(0,11).
0
1
2
3
4
5
6
7
8
9
11
ok
4> cau05:lietkedoixung(120,130).
121
ok
5> cau05:lietkedoixung(441,500).
444
454
464
474
484
494
ok
```

Câu 6.

- Code:

```
1 -module(cau06).
2 -export([dao/1, gtri/1, daoNguoc/1]).
3
4 dao(0) -> [];
5 dao(N) -> [N rem 10 | dao(N div 10)].
6
7 gtri([]) -> 0;
8 gtri([H|T]) -> gtri(T) * 10 + H.
9
10 daoNguoc(N) ->
11   gtri(lists:reverse(dao(N))).
```

- Giải thích:
 - Hàm dao(N) trả về list chứa các chữ số của N sau khi đảo ngược.
 - Hàm reverse (thuộc module lists) trả về đảo của list.
 - Hàm daoNguoc(N) trả về giá trị đảo ngược của số N.
- Kết quả:

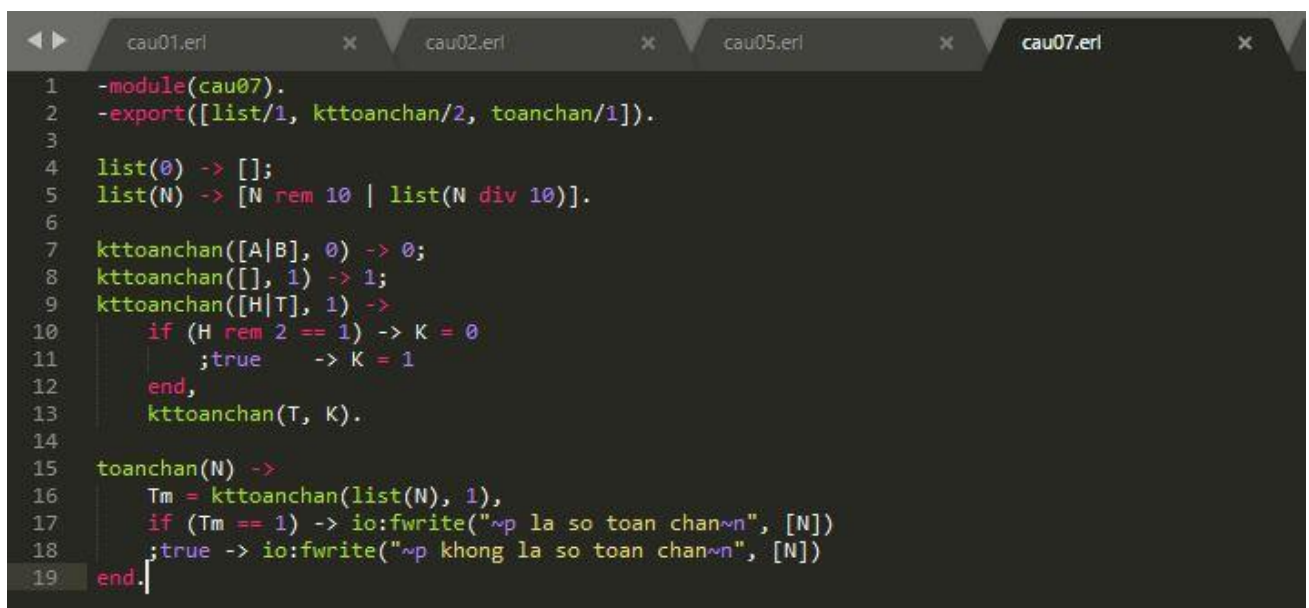
```

Eshell V9.1 (abort with ^G)
1> cd("/Erlang_project").
E:/Erlang/usr
ok
2> c(cau06).
{error,non_existing}
3> cd("/Erlang_project").
E:/Erlang_project
ok
4> c(cau06).
{ok,cau06}
5> cau06:daoNguoc(65481).
18456
6> cau06:daoNguoc(200).
2
7> █

```

Câu 7.

- Code



```

1  -module(cau07).
2  -export([list/1, kttoanchan/2, toanchan/1]).
3
4  list(0) -> [];
5  list(N) -> [N rem 10 | list(N div 10)].
6
7  kttoanchan([A|B], 0) -> 0;
8  kttoanchan([], 1) -> 1;
9  kttoanchan([H|T], 1) ->
10     if (H rem 2 == 1) -> K = 0
11     ;true -> K = 1
12     end,
13     kttoanchan(T, K).
14
15  toanchan(N) ->
16     Tm = kttoanchan(list(N), 1),
17     if (Tm == 1) -> io:fwrite("~p la so toan chan~n", [N])
18     ;true -> io:fwrite("~p khong la so toan chan~n", [N])
19  end.

```

- Giải thích:
 - Hàm list(N) trả về list chứa các chữ số của N sau khi đảo ngược.
 - Hàm kttoanchan kiểm tra 1 danh sách có phải toàn chữ số chẵn hay không, nếu phải trả về 1, ngược lại trả về 0.
 - Hàm toanchan(N) in ra sự toàn chẵn của N.
- Kết quả:


```
Erlang
File Edit Options View Help
494
ok
6> c(cau07).
cau07.erl:7: Warning: variable 'A' is unused
cau07.erl:7: Warning: variable 'B' is unused
{ok,cau07}
7> cau07:toanchan(24600084260).
24600084260 la so toan chan
ok
8> cau07:toanchan(246000842160).
246000842160 khong la so toan chan
ok
9> cau07:toanchan(2459226000842160).
2459226000842160 khong la so toan chan
ok
```

Câu 8.

- Code:

```
cau06.erl x cau08.erl x
1 | module(cau08).
2 | -export([chu_so/1, ktle/2, co_le/1]).
3 |
4 | chu_so(0) -> [];
5 | chu_so(N) -> [N rem 10 | chu_so(N div 10)].
6 |
7 | ktle(_, 0) -> 0;
8 | ktle([], 1) -> 1;
9 | ktle([H|T], 1) ->
10 |     if (H rem 2 == 1) -> K = 0
11 |         ;true -> K = 1
12 |     end,
13 |     ktle(T, K).
14 |
15 | co_le(N) ->
16 |     Tmp = ktle(chu_so(N), 1),
17 |     if (Tmp == 0) -> io:fwrite("~p ton tai chu so le~n", [N])
18 |         ;true -> io:fwrite("~p khong ton tai chu so le~n", [N])
19 | end.
```

- Giải thích:
 - Hàm chu_so(N) trả về list chứa các chữ số của N sau khi đảo ngược.
 - Hàm ktle kiểm tra tính lẻ của các phần tử trong list.
 - Hàm co_le(N) in ra có tồn tại chữ số lẻ trong số N hay không.
- Kết quả:

```

7> c(cau08).
{ok,cau08}
8> cau08:co_le(165489).
165489 ton tai chu so le
ok
9> cau08:co_le(2628064).
2628064 khong ton tai chu so le
ok
10> cau08:co_le(0).
0 khong ton tai chu so le
ok
11> cau08:co_le(3).
3 ton tai chu so le
ok
12> █

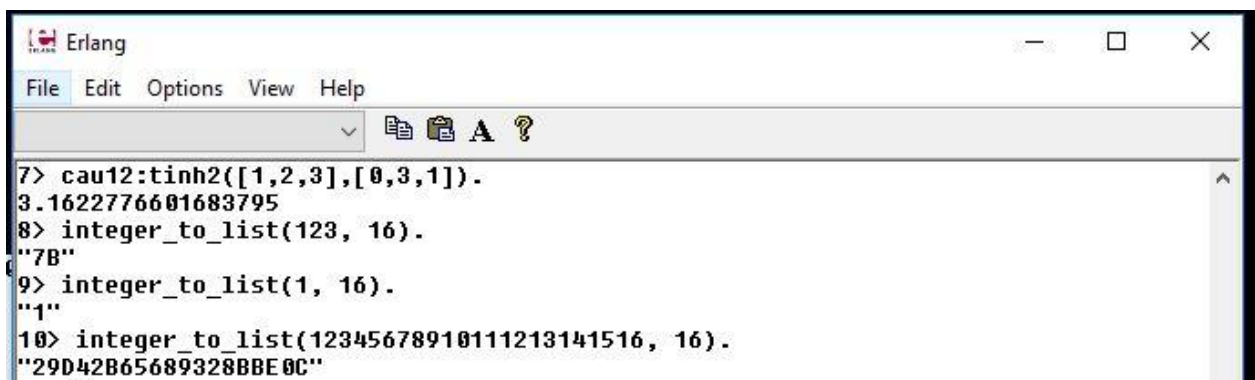
```

Câu 9.

- Code:

Có sẵn hàm chuyển đổi, không cần code

- Kết quả:



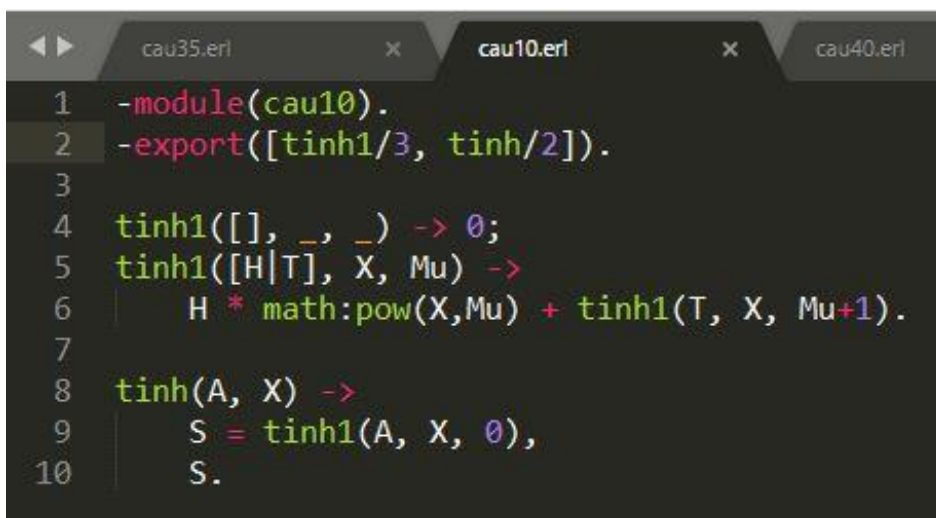
```

Erlang
File Edit Options View Help
7> cau12:tinh2([1,2,3],[0,3,1]).
3.1622776601683795
8> integer_to_list(123, 16).
"7B"
9> integer_to_list(1, 16).
"1"
10> integer_to_list(12345678910111213141516, 16).
"29D42B65689328BBE0C"

```

Câu 10.

- Code:



```

cau35.erl x cau10.erl x cau40.erl
1 -module(cau10).
2 -export([tinh1/3, tinh/2]).
3
4 tinh1([], _, _) -> 0;
5 tinh1([H|T], X, Mu) ->
6     H * math:pow(X,Mu) + tinh1(T, X, Mu+1).
7
8 tinh(A, X) ->
9     S = tinh1(A, X, 0),
10     S.

```

- Giải thích:

- Hàm `tinh1(Mang, X, Mu)` tính đa thức theo yêu cầu đề bài.
- Hàm `tinh(Mang, X)` in ra kết quả.
 - Kết quả:

```
Erlang
File Edit Options View Help
37> cau10:tinh([3,2,4,5],2).
63.0
38> c(cau10).
{ok,cau10}
39> cau10:tinh([3,2,4,5],2).
63.0
40> cau10:tinh([3,1,3,4,0],7).
1529.0
41> cau10:tinh([3,1,3,4,5,8],1).
24.0_
```

Câu 11.

- Code:

```
cau11.erl
1 -module(cau11).
2 -export([min/2, tinh1/3, tinh2/3]).
3
4 min(X, Y) ->
5     if X < Y -> X;
6         true -> Y
7     end.
8
9 tinh1(_, [], _) -> 123456789;
10 tinh1(X, [H|T], Q) ->
11     if X + H > Q -> S = X + H;
12         true -> S = 123456789
13     end,
14     min(S, tinh1(X, T, Q)).
15
16 tinh2([], _, _) -> 123456789;
17 tinh2([H|T], L, Q) ->
18     min(tinh1(H, L, Q), tinh2(T, L, Q)).
```

- Giải thích:
 - Hàm `min(X, Y)` trả về min của 2 số X, Y.
 - Hàm `tinh1(X, L, Q)` trả về tổng nhỏ nhất của X với phần tử nào đó trong L thỏa tổng lớn hơn Q.
 - Hàm `tinh2(A, B, Q)` trả về tổng nhỏ nhất của phần tử nào đó trong A với phần tử nào đó trong B thỏa tổng lớn hơn Q.
- Kết quả:

```
51> cau11:tinh2([1, 2, 3, 4, 4, 7, 6, 8], [2, 6, 8, 7, 6], 10).
11
52> cau11:tinh2([1, 8], [2, 6, 8, 7, 6], 10).
14
53> cau11:tinh2([1, 8], [2, 6, 8, 7, 6], 3).
7
_
```

Câu 12.

- Code:

```

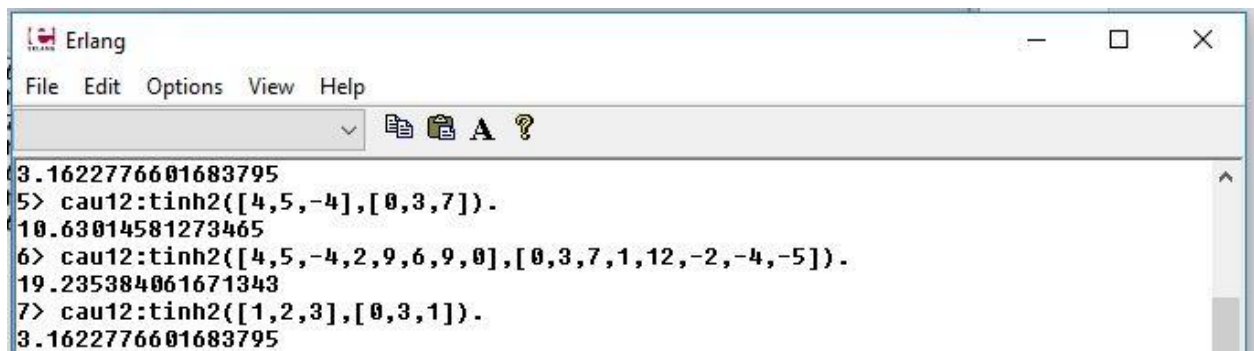
1 -module(cau12).
2 -export([max/2, tinh1/4, tinh2/2]).
3
4 max(A, B) ->
5     if
6         (A < B) -> B
7         ;true -> A
8     end.
9
10 tinh1([], [], _, _) -> 0;
11 tinh1([Dau|Dui], [H|T], X, Y) ->
12     Kc = math:sqrt((X-Dau)*(X-Dau) + (Y-H)*(Y-H)),
13     max(Kc, tinh1(Dui, T, X, Y)).
14
15 tinh2([], []) -> 0;
16 tinh2([Dau|Dui], [H|T]) ->
17     max(tinh1(Dui, T, Dau, H), tinh2(Dui, T)).

```

- Giải thích:

- Hàm max(A, B) trả về giá trị lớn hơn giữa 2 số A và B.
- Hàm tinh1(Mangx, Mangy, X, Y) trả về giá trị lớn nhất của khoảng cách giữa điểm (X, Y) với các điểm thuộc (Mangx, Mangy).
- Hàm tinh2(Mangx, Mangy) trả về khoảng cách lớn nhất giữa 2 điểm thuộc tập Mangx, Mangy.

- Kết quả:



```

Erlang
File Edit Options View Help
3.1622776601683795
5> cau12:tinh2([4,5,-4],[0,3,7]).
10.63014581273465
6> cau12:tinh2([4,5,-4,2,9,6,9,0],[0,3,7,1,12,-2,-4,-5]).
19.235384061671343
7> cau12:tinh2([1,2,3],[0,3,1]).
3.1622776601683795

```

Câu 13.

- Code:

```

1 -module(cau13).
2 -export([nearMax/1]).
3
4 nearMax(L) ->
5     Len = length(L),
6     if (Len < 2) -> io:fwrite("khong co so lon thu hai~n")
7     ;true ->
8         L1 = lists:reverse(lists:usort(L)),
9         lists:nth(2,L1)
10     end.

```

- Giải thích:
 - Hàm length trả về độ dài của list.
 - Nếu độ dài < 2 thì không tìm được số lớn thứ 2 trong list.
 - Nếu > 2 thì L1 trả về list đã sắp xếp giảm dần và xóa các phần tử trùng nhau của list ban đầu. Ta chỉ cần lấy phần tử thứ 2 của L1 sẽ được kết quả.
- Kết quả:

```
7> c(cau13).
{ok,cau13}
8> cau13:nearMax([0]).
khong co so lon thu hai
ok
9> cau13:nearMax([1,2,2,-3,6,9,5,5,9]).
6
10> cau13:nearMax([1,2]).
1
11> █
```

Câu 14.

- Code:

```
collection.brt x cau14.erl x
1 | module(cau14).
2 | -export([ktra/3,find/3]).
3 |
4 | ktra([],_,_) -> io:fwrite("khong co so nao thoa dieu kien~n");
5 | ktra([H|T],X,Y) ->
6 |     if H <= Y, H >= X -> H
7 |     ; true -> ktra(T,X,Y)
8 |     end.
9 |
10 | find(L,X,Y) ->
11 |     L1 = lists:usort(L),
12 |     ktra(L1,X,Y).
```

- Giải thích:
 - Hàm ktra(L,X,Y) kiểm tra trong list L có phần tử nào nhỏ nhất thuộc đoạn [X,Y].
 - Hàm find để trả về kết quả.
- Kết quả:

```
11> c(cau14).
{ok,cau14}
12> cau14:find([1,2,3,4],1,4).
1
13> cau14:find([1,2,3,4],1.8,4).
2
14> cau14:find([1,2,3,4],4.2,6).
khong co so nao thoa dieu kien
ok
15> █
```

Câu 15.

- Code:

```

1  -module(cau15).
2  -export([sochuso/1, pow/2, tinh/2, isAmstrong/1]).
3
4  sochuso(N) ->
5      if 0 <= N, N <= 9
6          -> 1;
7      true
8          -> 1 + sochuso(N div 10)
9      end.
10
11 pow(_, 0) -> 1;
12 pow(X, Y) ->
13     T = pow(X, Y div 2),
14     if Y rem 2 /= 0
15         -> T * T * X;
16     true
17         -> T * T
18     end.
19
20 tinh(0, _) -> 0;
21 tinh(X, Y) ->
22     pow(X rem 10, Y) + tinh(X div 10, Y).
23
24 isAmstrong(X) ->
25     case tinh(X, sochuso(X)) of
26         X -> io:fwrite("~p la so Amstrong~n", [X]);
27         _ -> io:fwrite("~p ko la so Amstrong~n", [X])
28     end.

```

- Giải thích:
 - Hàm sochuso(N) cho biết số chữ số của N.
 - Hàm pow(X, Y) tính X^Y .
 - Hàm tinh(X, Y) tính tổng mũ N các chữ số của X (N là số chữ số của X).
 - Hàm isAmstrong(X) kiểm tra X có là số Amstrong không.
- Kết quả:

```

8> cau15:isAmstrong(153).
153 la so Amstrong
ok
9> cau15:isAmstrong(123).
123 ko la so Amstrong
ok
10> cau15:isAmstrong(1).
1 la so Amstrong
ok
11> cau15:isAmstrong(123456789101112131141516).
123456789101112131141516 ko la so Amstrong
ok

```

Câu 16.

- Code:

```

1 -module(cau16).
2 -import(lists,[member/2]).
3 -export([thuocAkoB/2]).
4
5 thuocAkoB([], _) -> io:fwrite("");
6 thuocAkoB([H|T], B) ->
7     Status = member(H, B),
8     if
9         Status == true -> thuocAkoB(T, B)
10    ;true -> io:fwrite("~p~n", [H]), thuocAkoB(T, B)
11 end.

```

- Giải thích:
 - Hàm `thuocAkoB(A, B)` trả về những phần tử có trong A mà không có trong B.
- Kết quả:

```

Erlang
File Edit Options View Help
ok
39> cau16:thuocAkoB([1,2,3,4,5,6,7,8,9],[4,5,6,7,89,0,2]).
1
3
8
9
ok
40> cau16:thuocAkoB([2,4,5,6,7,42],[4,5,6,7,89,0,2]).
42
ok
41> cau16:thuocAkoB([2,4,5,6,7,69,69,69],[4,5,6,7,89,0,2]).
69
69
69
ok _

```

Câu 17.

- Code:

```

1 -module(cau17).
2 -import(lists,[usort/1, append/2]).
3 -export([ghep/2]).
4
5 ghep(A, B) ->
6     C = append(A, B),
7     usort(C).

```

- Giải thích:
 - Hàm `ghep(A, B)` nối list B với A, dùng hàm `usort` có trong thư viện `lists` để sắp xếp lại C đồng thời loại bỏ phần tử trùng nhau.
- Kết quả:

```
Erlang
File Edit Options View Help
19> c(cau17).
{ok,cau17}
20> cau17:ghep([3,2,1],[2]).
[1,2,3]
21> cau17:ghep([3,2,1],[9,8,3]).
[1,2,3,8,9]
22> cau17:ghep([1,2,3,4,5,6,6,7],[9,8,3,6,9]).
[1,2,3,4,5,6,7,8,9]
```

Câu 18.

- Code:

```
cau18.erl
1 -module(cau18).
2 -import(lists, [reverse/1, splitwith/2]).
3 -export([dao/1]).
4
5 dao([]) -> [];
6 dao(L) ->
7     [H|T] = L,
8     if H == $ ->
9         " " ++ dao(T);
10    true ->
11        {X, Y} = splitwith(fun(A) -> A /= $ end, L),
12        reverse(X) ++ dao(Y)
13    end.
14
```

- Giải thích:

- Dùng hàm splitwith để tách phần đầu đến kí tự khoảng trắng sau đó dùng hàm reverse để đảo và thực hiện tiếp tục với phần còn lại.

- Kết quả:

```
48> c(cau18).
{ok,cau18}
49> cau18:dao("anh di hoc").
"hna id coh"
50> cau18:dao("anh ko hoc").
"hna ok coh"
51> cau18:dao("anh di ngu").
"hna id ugn"
52> cau18:dao("anh met vs cai erlang nay qua").
"hna_tem sv iac gnalre yan auq"
```

Câu 19.

- Code:


```

1 -module(cau19).
2 -export([isPrime/1, prime/2, takePrime/1, locate_MaxPrime/1]).
3
4 % Ham ktra so nguyen to
5 prime(N,A)
6     when A*A > erlang:abs(N) -> true;
7 prime(N,M)->
8     Du = N rem M,
9     if Du == 0 -> false
10        ;true -> prime(N,M+1)
11    end.
12 isPrime(-1)-> false;
13 isPrime(0)-> false;
14 isPrime(1)-> false;
15 isPrime(2)-> true;
16 isPrime(N) -> prime(N,2).
17
18 takePrime([]) -> [];
19 takePrime([H|T]) ->
20     Check = isPrime(H),
21     if Check == true -> takePrime(T) ++ [H]
22        ;true -> takePrime(T)
23    end.
24
25 % main
26 locate_MaxPrime(List) ->
27     Len = length(takePrime(List)),
28     if Len < 1 -> -1
29        ;true ->
30            L = lists:usort(takePrime(List)),
31            S = lists:last(L),
32            string:str(List,[S])
33    end.
34

```

- Giải thích:
 - Hàm takePrime lấy list các số nguyên tố từ 1 list cho trước.
 - Nếu takePrime ko có phần tử nào thì trả về -1.
 - Nếu có thì dùng hàm str lấy vị trí của phần tử max trong list nguyên tố.
- Kết quả:

```

16> c(cau19).
{ok,cau19}
17> cau19:locate_MaxPrime([1,2,3,9,11,3,6,5]).
5
18> cau19:locate_MaxPrime([1,3,9,3,6]).
2
19> cau19:locate_MaxPrime([1,4,9,6]).
-1
_

```

Câu 20.

- Code:

```

1 -module(cau20).
2 -export([tinhkeyvalue/2, maxa/2, tinh/4, tim/2]).
3
4 tinhkeyvalue([], _) -> [];
5 tinhkeyvalue([H|T], I) ->
6   [{H, I}] ++ tinhkeyvalue(T, I + 1).
7
8 maxa({Ai1, I1, J1}, {Ai2, I2, J2}) ->
9   if Ai1 > Ai2 ->
10    {Ai1, I1, J1};
11   true ->
12    {Ai2, I2, J2}
13   end.
14
15 tinh([], _, _, _) -> {-123456, 0, 0};
16 tinh([H|T], I, K, Map) ->
17   J = maps:get(K - H, Map, -1),
18   if J == -1 ->
19    tinh(T, I + 1, K, Map);
20   true ->
21    maxa({H, I, J}, tinh(T, I + 1, K, Map))
22   end.
23
24 tim(L, K) ->
25   Map = maps:from_list(tinhkeyvalue(L, 1)),
26   {_, I, J} = tinh(L, 1, K, Map),
27   io:fwrite("~p ~p~n", [I, J]).

```

- Giải thích:
 - Hàm tinhkeyvalue đưa list ban đầu về dạng [{Key, Value}].
 - Sau đó dùng map để đánh dấu giá trị các phần tử. Rồi với mỗi I, dùng map để tìm J tương ứng thỏa điều kiện.
- Kết quả:

```

68> c(cau20).
{ok,cau20}
69> cau20:tim([1, 2, 7, 8, 9, -3, 0, 6], 13).
3 8
ok
70> cau20:tim([1, 2, 7, 8, 9, -3, 0, 6, 16], 13).
9 6
ok
71> cau20:tim([100000, 2, 7, 8, 9, -3, 0, 6, 16, -100000], 0).
1 10
ok

```

Câu 21.

- Code:

```

1 -module(cau21).
2 -export([lon_thu_k/2]).
3
4 lon_thu_k(A,K) ->
5   L = lists:reverse(lists:usort(A)),
6   Len = length(L),
7   if Len < K -> io:fwrite("khong co phan tu lon thu ~p~n",[K])
8   ;true -> lists:nth(K,L)
9   end.
10

```

- Giải thích:
 - L trả về list A sau khi sắp xếp giảm dần và xóa các phần tử trùng nhau.
 - Phần tử thứ K trong L chính là phần tử lớn thứ K trong list A.
- Kết quả:

```

20> c(cau21).
{ok,cau21}
21> cau21:lon_thu_k([1,8,3,9,2,0,-5,7],3).
7
22> cau21:lon_thu_k([1,8,3,9,2,0,-5,7],1).
9
23> cau21:lon_thu_k([8,3,9,7,7],4).
3
24> cau21:lon_thu_k([8,3,9,7,7],5).
khong co phan tu lon thu 5
ok
25> █

```

Câu 22.

- Code:

```

1 -module(cau22).
2 -import(maps, [get/2, from_list/1]).
3 -export([tinhkeyvalue/2, min/2, tinh1/6, tinh2/4, vitri/2]).
4
5 tinhkeyvalue([], _) -> [];
6 tinhkeyvalue([H|T], I) ->
7     [{H,I}] ++ tinhkeyvalue(T, I+1).
8
9 min({Val1, I1, J1}, {Val2, I2, J2}) ->
10     if
11         Val1 < Val2 -> {Val1, I1, J1}
12         ;true       -> {Val2, I2, J2}
13     end.
14
15 tinh1(X, [], Val, I, Map, AD) ->
16     if
17         AD > 0 -> J = get(Val-X, Map)
18         ;true  -> J = get(-Val-X, Map)
19     end,
20     {Val, I, J};
21 tinh1(X, [H|T], CurVal, I, Map, AD) ->
22     NewVal = erlang:abs(X+H),
23     NAD = X+H,
24     if
25         NewVal < CurVal -> tinh1(X, T, NewVal, I, Map, NAD)
26         ;true           -> tinh1(X, T, CurVal, I, Map, AD)
27     end.
28
29 tinh2([], _, _, _) -> {987654, 0, 0};
30 tinh2([H|T], B, I, Map) ->
31     {X, Y, Z} = tinh1(H, B, 987654, I, Map, 0),
32     {M, N, P} = erlang:min({X, Y, Z}, tinh2(T, B, I+1, Map)),
33     {M, N, P}.
34
35 vitri(A, B) ->
36     Map = from_list(tinhkeyvalue(B, 1)),
37     {_, I, J} = tinh2(A, B, 1, Map),
38     io:fwrite("~p ~p~n", [I, J]).

```

- Giải thích:

- Hàm tinhkeyvalue lưu giá trị và thứ tự của giá trị trong 1 list vào 1 list.
- Hàm min tính giá trị nhỏ hơn giữa 2 giá trị.
- Hàm tinh1 tính giá trị tuyệt đối của X với từng giá trị trong list B, trả về giá trị nhỏ nhất và vị trí của phần tử đưa về giá trị đó trong B.
- Hàm tinh2 tính trị tuyệt đối của tổng nhỏ nhất, trả về giá trị đó và vị trí của nó trong 2 list.
- Hàm vitri(A, B) in ra vị trí 2 giá trị có trị tuyệt đối tổng nhỏ nhất trong 2 list A và B.

- Kết quả:

```
Erlang
File Edit Options View Help
cau22.erl:31: function tinh1/5 undefined
error
72> c(cau22).
{ok,cau22}
73> cau22:vitri([3,2,1],[2,-3,7]).
1 2
ok
74> cau22:vitri([1,35,1],[2,-34,7]).
2 2
ok
75> cau22:vitri([-69,35,0],[2,-34,69]).
1 3
ok
```

Câu 23.

- Code:

```
cau23.erl
1 -module(cau23).
2 -import(lists, [reverse/1, splitwith/2]).
3 -export([dao/1]).
4
5 dao([]) -> [];
6 dao(L) ->
7     [H|T] = L,
8     if H == $ ->
9         dao(T) ++ " ";
10    true ->
11        {X, Y} = splitwith(fun(A) -> A /= $ end, L),
12        dao(Y) ++ X
13    end.
14
```

- Giải thích:

- Tách từ đầu tiên ra, đảo phần còn lại rồi ghép từ vừa tách vào cuối.

- Kết quả:

```
5> c(cau23).
{ok,cau23}
6> cau23:dao("anh đi học").
"học đi anh"
7> cau23:dao("anh không học").
"học không anh"
8> cau23:dao("anh đi ngủ").
"ngủ đi anh"
9> cau23:dao("anh met vs cai erlang nay qua").
"qua_nay erlang cai vs met anh"
```

Câu 24.

- Code:

```

1  -module(cau24).
2  -import(lists, [reverse/1, splitwith/2]).
3  -export([viet/1, viethoadau/1]).
4
5  viethoadau([H|T]) ->
6      if $a <= H, H <= $z ->
7          [H + $A - $a] ++ T;
8      true ->
9          [H|T]
10     end.
11
12 viet([]) -> [];
13 viet(L) ->
14     [H|T] = L,
15     if H == $ ->
16         " " ++ viet(T);
17     true ->
18         {X, Y} = splitwith(fun(A) -> A /= $ end, L),
19         viethoadau(X) ++ viet(Y)
20     end.
21

```

- Giải thích:
 - Tách từ đầu tiên ra, viết hoa chữ cái đầu, thực hiện tiếp với phần còn lại rồi ghép vô.
- Kết quả:

```

14> c(cau24).
{ok,cau24}
15> cau24:viet("Anh đi học").
"Anh đi học"
16> c(cau24).
{ok,cau24}
17> cau24:viet("Anh đi học").
"Anh Đi Học"
18> cau24:viet("Anh ko học").
"Anh Ko Học"
19> cau24:viet("anh đi Ngu").
"Anh Đi Ngu"
20> cau24:viet("Anh met vs cai Erlang nay qua?").
"Anh_Met Vs Cai Erlang Nay Qua?"

```

Câu 25.

- Code:

```

1  -module(cau25).
2  -export([isPrime/1,prime/2,takeNonPrime/1,trich/2]).
3
4  % Ham ktra so nguyen to
5  prime(N,A)
6  |   when A*A > erlang:abs(N) -> true;
7  √ prime(N,M)->
8  |   Du = N rem M,
9  |   if Du == 0 -> false
10 |   ;true -> prime(N,M+1)
11 |   end.
12 isPrime(-1)-> false;
13 isPrime(0)-> false;
14 isPrime(1)-> false;
15 isPrime(2)-> true;
16 isPrime(N) -> prime(N,2).
17
18 takeNonPrime([]) -> [];
19 √ takeNonPrime([H|T]) ->
20 |   Check = isPrime(H),
21 |   if Check == false -> [H]++takeNonPrime(T)
22 |   ;true -> takeNonPrime(T)
23 |   end.
24
25 √ trich(A,B) ->
26 |   L = takeNonPrime(A),
27 |   L++B.
28

```

- Giải thích:
 - Hàm takeNonPrime trả về list các số không nguyên tố từ một list cho trước.
 - Hàm trich(A,B) lấy các số không nguyên tố từ A cho vào list B.
- Kết quả:

```

16> c(cau25).
{ok,cau25}
17> cau25:trich([1,2,3,4],[0]).
[1,4,0]
18> cau25:trich([1,2,3,4,-1],[0,-1,2]).
[1,4,-1,0,-1,2]
19> cau25:trich([1,2,3,4,-11],[0,-2]).
[1,4,0,-2]
20> █

```

Câu 26.

- Code:

```

1 -module(cau26).
2 -export([isPrime/1, prime/2, delPrime/1]).
3
4 % Ham ktra so nguyen to
5 prime(N,A)
6     when A*A > erlang:abs(N) -> true;
7 prime(N,M)->
8     Du = N rem M,
9     if Du == 0 -> false
10        ;true -> prime(N,M+1)
11    end.
12 isPrime(-1)-> false;
13 isPrime(0)-> false;
14 isPrime(1)-> false;
15 isPrime(2)-> true;
16 isPrime(N) -> prime(N,2).
17
18 delPrime([]) -> [];
19 delPrime([H|T]) ->
20     L = [H|T],
21     Check = isPrime(H),
22     if Check == true -> delPrime(lists:delete(H,L))
23        ;true -> [H]++delPrime(T)
24    end.
25

```

- Giải thích:
 - Hàm delete(H,L) trả về list sau khi xóa H khỏi list L.
 - Hàm delPrime(L) trả về list sau khi đã xóa các số nguyên tố có trong list L ban đầu.
- Kết quả:

```

22> c(cau26).
{ok,cau26}
23> cau26:delPrime([1,2,3,4,-11]).
[1,4]
24> cau26:delPrime([1,2,3,4,0,-1]).
[1,4,0,-1]
25> cau26:delPrime([1,2,3,4,0,-7]).
[1,4,0]
26>

```

Câu 27.

- Code:

```

1 -module(cau27).
2 -export([layptu/1, xuly/1]).
3
4 layptu([]) -> [];
5 layptu([H|T]) ->
6     if
7         H rem 10 == 6, H rem 6 == 0 -> layptu(T) ++ [H]
8         ;true -> layptu(T)
9     end.
10
11 xuly(A) ->
12     io:format("~p phan tu tan cung la 6, chia het cho 6~n", [length(layptu(A))]).

```

- Giải thích:
 - Hàm layptu(A) lấy những phần tử tận cùng là 6, chia hết cho 6 bỏ vào list.

- Hàm xuly(A) in ra số phần tử của list.
 - Kết quả:

```

Erlang
File Edit Options View Help
{ok,cau27}
14> cau27:xuly([36,66,6,12,18]).
3 phần tử tan cùng là 6, chia hết cho 6
ok
15> cau27:xuly([36,66,6,12,18,24,1,1,2,3,4]).
3 phần tử tan cùng là 6, chia hết cho 6
ok
16> cau27:xuly([12,18,24,1,1,2,3,4,69]).
0 phần tử tan cùng là 6, chia hết cho 6
ok
_

```

Câu 28.

- Code:

```

1 -module(cau28).
2 -export([isPrime/1,prime/2,takePrime/1,tinh/1]).
3
4 % Ham ktra so nguyen to
5 prime(N,A)
6   when A*A > erlang:abs(N) -> true;
7 prime(N,M)->
8   Du = N rem M,
9   if Du == 0 -> false
10    ;true -> prime(N,M+1)
11   end.
12 isPrime(-1)-> false;
13 isPrime(0)-> false;
14 isPrime(1)-> false;
15 isPrime(2)-> true;
16 isPrime(N) -> prime(N,2).
17
18 % Lay danh sach cac so nguyen to
19 takePrime([]) -> [];
20 takePrime([H|T]) ->
21   Check = isPrime(H),
22   if Check == true -> takePrime(T) ++ [H]
23    ;true -> takePrime(T)
24   end.
25
26 tinh(L) ->
27   L1 = takePrime(L),
28   Len = length(L1),
29   if Len < 1 -> 0
30    ;true -> lists:sum(L1) / Len
31   end.

```

- Giải thích:
 - Trong hàm tinh(L): Lấy list các số nguyên tố trong L, trả về tổng của các số nguyên tố này chia cho số lượng của chúng. Ta được trung bình cộng các số nguyên tố trong L.
- Kết quả:

```

26> c(cau28).
{ok,cau28}
27> cau28:tinh([1,2,3,4,0,-7]).
-0.6666666666666666
28> cau28:tinh([1,2,3,4,0,-1]).
2.5
29> cau28:tinh([1,4,0,-1]).
0
30> █

```

Câu 29.

- Code:

```

1  -module(cau29).
2  -export([isPrime/1,prime/2,takePrime/1,countPrime/1]).
3
4  % Ham ktra so nguyen to
5  prime(N,A)
6  |   when A*A > erlang:abs(N) -> true;
7  prime(N,M)->
8  |   Du = N rem M,
9  |   if Du == 0 -> false
10 |   ;true -> prime(N,M+1)
11 |   end.
12 isPrime(-1)-> false;
13 isPrime(0)-> false;
14 isPrime(1)-> false;
15 isPrime(2)-> true;
16 isPrime(N) -> prime(N,2).
17
18 % Lay danh sach cac so nguyen to
19 takePrime([]) -> [];
20 takePrime([H|T]) ->
21 |   Check = isPrime(H),
22 |   if Check == true -> takePrime(T) ++ [H]
23 |   ;true -> takePrime(T)
24 |   end.
25
26 countPrime(L) ->
27 |   L1 = lists:usort(takePrime(L)),
28 |   length(L1).

```

- Giải thích:

- Lấy list các số nguyên tố trong list L rồi xóa các phần tử trùng nhau, gán list này cho L.
- Trả về độ dài của L ta được kết quả.

- Kết quả:

```

30> c(cau29).
{ok,cau29}
31> cau29:countPrime([1,2,3,4,5,6]).
3
32> cau29:countPrime([1,2,3,4,5,6,-1]).
3
33> cau29:countPrime([1,2,3,4,5,6,-11]).
4
34> cau29:countPrime([1,2,3,4,5,6,-2]).
4
35> █

```

Câu 30.

- Code:

```

1  -module(cau30).
2  -export([inbit/2, nhiphan/1]).
3
4  inbit(_, -1) -> io:fwrite("~n");
5  inbit(X, I) ->
6      io:fwrite("~p", [((X bsr I) band 1)]),
7      inbit(X, I - 1).
8
9  nhiphan(X) ->
10     inbit(X, 15).

```

- Giải thích: Dùng biểu thức $((X \text{ bsr } I) \text{ band } 1)$ để lấy bit thứ I của X và in lần lượt các bit ra.
- Kết quả:

```

39> c(cau30).
{ok,cau30}
40> cau30:nhiphan(10).
0000000000001010
ok
41> cau30:nhiphan(-10).
1111111111110110
ok
42> cau30:nhiphan(255).
0000000011111111
ok
43> cau30:nhiphan(-255).
1111111100000001
ok
44> cau30:nhiphan(-8).
1111111111111000
ok

```

Câu 31.

- Code:

```

1  -module(cau31).
2  -export([thapphan/1]).
3
4  gtri(0, _) -> 0;
5  gtri(X, I) ->
6      (X rem 10) * (1 bsl I) + gtri(X div 10, I + 1).
7
8  thapphan(X) ->
9      T = X rem 1000000000000000,
10     -(X div 1000000000000000) * (1 bsl 15) + gtri(T, 0).

```

- Giải thích: Vì có 16 bit nên nếu bit ngoài cùng là 0 thì tính giá trị số dương còn nếu là 1 thì tính 15 bit còn lại theo giá trị dương rồi trừ cho 2^{15} .
- Kết quả:

```

74> c(cau31).
{ok,cau31}
75> cau31:thapphan(1111111111111000).
-8
76> cau31:thapphan(0000000000001000).
8
77> cau31:thapphan(111111111111111).
-1
78> cau31:thapphan(0000000000000000).
0
79> cau31:thapphan(111111100000001).
-255
80> cau31:thapphan(0000000011111111).
255

```

Câu 32.

- Code:

```

1 -module(cau32).
2 -export([tinhF/1]).
3
4 tinhF(1) -> 1;
5 tinhF(2) -> 2;
6 tinhF(N) ->
7     N * N / (N-1) * tinhF(N-1).
8

```

- Giải thích:
 - Từ đề bài ta suy ra được $f(n) = n^2 / (n-1) * f(n-1)$.
 - Dùng hàm đệ quy tinhF(N) để tính f(n).
- Kết quả:

```

35> c(cau32).
{ok,cau32}
36> cau32:tinhF(2).
2
37> cau32:tinhF(6).
2160.0

```

Câu 33.

- Code:

```

1  -module(cau33).
2  -export([isPrime/1,prime/2,takePrime/1,tong/1]).
3
4  % Ham ktra so nguyen to
5  prime(N,A)
6      when A*A > erlang:abs(N) -> true;
7  prime(N,M)->
8      Du = N rem M,
9      if Du == 0 -> false
10         ;true -> prime(N,M+1)
11     end.
12 isPrime(-1)-> false;
13 isPrime(0)-> false;
14 isPrime(1)-> false;
15 isPrime(2)-> true;
16 isPrime(N) -> prime(N,2).
17
18 % Lay danh sach cac so nguyen to
19 takePrime([]) -> [];
20 takePrime([H|T]) ->
21     Check = isPrime(H),
22     if Check == true -> takePrime(T) ++ [H]
23         ;true -> takePrime(T)
24     end.
25
26 tong(L) ->
27     L1 = takePrime(L),
28     Len = length(L1),
29     if Len < 1 -> 0
30         ;true -> lists:sum(L1)
31     end.

```

- Giải thích:
 - Lấy các số nguyên tố có trong L bằng hàm takePrime.
 - Sau đó tính tổng của chúng.
- Kết quả:

```

46> c(cau33).
{ok,cau33}
47> cau33:tong([1,4,6,8,9,0]).
0
48> cau33:tong([1,2,3,4,5,6,7,8,9,-2]).
15
49> cau33:tong([1,2,3,4,5,6,7,8,9,-1]).
17
50> cau33:tong([1,2,3,4,5,6,7,8,9]).
17

```

Câu 34.

- Code:

```

cau34.erl      x      cau37.erl      x      cau27.erl      x
1  -module(cau34).
2  -import(lists,[sort/1, reverse/1]).
3  -export([layduong/1, layam/1, lay0/1, sapxep/1]).
4
5  layduong([]) -> [];
6  layduong([H|T]) ->
7      if
8          H > 0 -> layduong(T) ++ [H]
9          ;true -> layduong(T)
10     end.
11
12 layam([]) -> [];
13 layam([H|T]) ->
14     if
15         H < 0 -> layam(T) ++ [H]
16         ;true -> layam(T)
17     end.
18
19
20 lay0([]) -> [];
21 lay0([H|T]) ->
22     if
23         H == 0 -> lay0(T) ++ [H]
24         ;true -> lay0(T)
25     end.
26
27 sapxep(A) ->
28     Sx = lay0(A) ++ reverse(sort(layduong(A))) ++ sort(layam(A)),
29     Sx.

```

- Giải thích:
 - Hàm layduong(A) lấy những phần tử dương bỏ vào list.
 - Hàm layam(A) lấy những phần tử âm bỏ vào list.
 - Hàm lay0(A) lấy những phần tử 0 bỏ vào list.
 - Hàm sapxep(A) sắp xếp theo yêu cầu đề bài.
- Kết quả:

```

Erlang
File Edit Options View Help
43> cau34:sapxep([35,-7,6,9,5,2,-1,-9,-100]).
[35,9,6,5,2,-100,-9,-7,-1]
44> c(cau34).
{ok,cau34}
45> cau34:sapxep([35,-7,6,9,5,2,0,-1]).
[0,35,9,6,5,2,-7,-1]
46> cau34:sapxep([35,-7,6,9,5,2,-1,-9,-100]).
[35,9,6,5,2,-100,-9,-7,-1]
47> cau34:sapxep([35,-7,6,9,5,2,-1,-9,-100,0,0,0,0,0,0,0,69]).
[0,0,0,0,0,0,0,69,35,9,6,5,2,-100,-9,-7,-1]

```

Câu 35.

- Code:

```

cau35.erl x cau40.erl x cau22.erl
1 -module(cau35).
2 -export([tim/3, itnhat/1]).
3
4 tim([], _, Val) -> Val;
5 tim(Lst, CurLen, CurVal) ->
6     [D|_] = Lst,
7     {Ht,C1} = lists:partition(fun(A) -> A == D end, Lst),
8     [H|_] = Ht,
9     NewLen = length(Ht),
10    if
11        NewLen < CurLen -> tim(C1, NewLen, H)
12        ;true           -> tim(C1, CurLen, CurVal)
13    end.
14
15 itnhat(L) ->
16     Min = tim(L,987654321,0),
17     io:fwrite("~p xuất hiện ít nhất~n", [Min]).

```

- Giải thích:
 - Hàm tim(Lst, CurLen, CurVal) lần lượt lấy hết các phần tử giống nhau trong list Lst, mỗi lần lấy so sánh số lượng với số lượng hiện tại, nếu nhỏ hơn thì lưu lại, cứ vậy đến khi hết list.
 - Hàm itnhat(L) sử dụng hàm tim tìm kiếm giá trị xuất hiện ít nhất và in kết quả.
- Kết quả:

```

Erlang
File Edit Options View Help
{ok,cau35}
27> cau35:itnhat([1,1,1,1,2,2,2,3,3,3,4,4,4,4,5,5]).
5 xuất hiện ít nhất
ok
28> cau35:itnhat([1,1,1,1,2,2,2,3,3,3,4,4,4,4,5,5,69]).
69 xuất hiện ít nhất
ok
29> cau35:itnhat([1,1,1,1,2,2,2,3,3,3,4,4,4,4,69,96,96,69,69]).
96 xuất hiện ít nhất
ok _

```

Câu 36.

- Code:

```

1 -module(cau36).
2 -import(lists,[split/2]).
3 -export([chen/3]).
4
5 chen(X, K, L1) ->
6     {L2, L3} = split(K-1, L1),
7     L2 ++ [X] ++ L3.

```

- Giải thích:
 - Hàm chen(X, K, L1) chèn phần tử X vào vị trí K bằng cách tách L1 ra thành 2 list và thêm X vào giữa L2, L3 trong list mới.
- Kết quả:

```

Erlang
File Edit Options View Help
error
52> c(cau36).
{ok,cau36}
53> cau36:chen(5,3,[1,1,1,1,1,1,1]).
[1,1,5,1,1,1,1]
54> cau36:chen(69,7,[1,1,1,1,1,1,1]).
[1,1,1,1,1,1,69,1]
55> cau36:chen(96,5,[1,1,1,1,1,1,1]).
[1,1,1,1,96,1,1,1]

```

Câu 37.

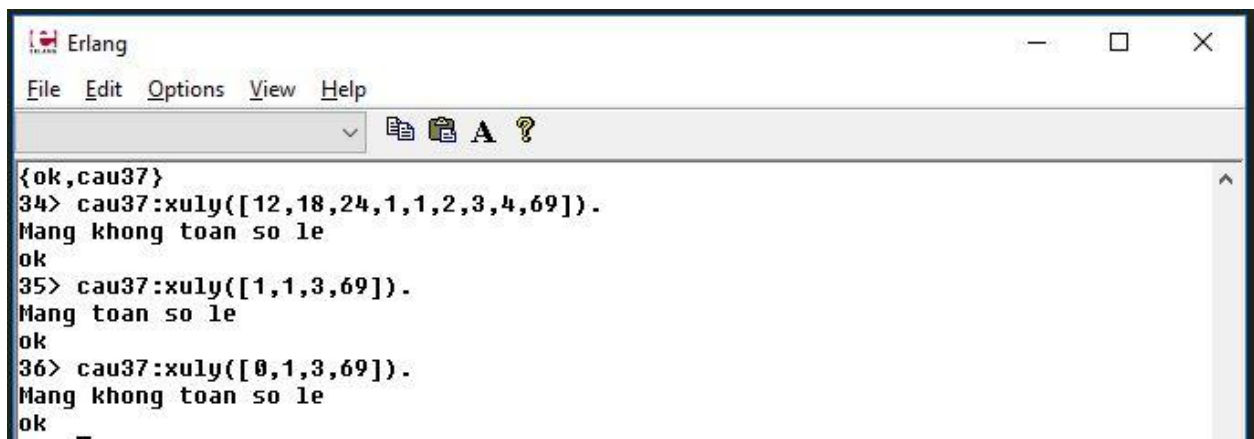
- Code:

```

1 -module(cau37).
2 -export([laysochan/1, xuly/1]).
3
4 laysochan([]) -> [];
5 laysochan([H|T]) ->
6     if
7         H rem 2 == 0 -> laysochan(T) ++ [H]
8         ;true -> laysochan(T)
9     end.
10
11 xuly(A) ->
12     Chan = length(laysochan(A)),
13     if
14         Chan == 0 ->
15         io:fwrite("Mang toan so le~n")
16         ;true -> io:fwrite("Mang khong toan so le~n")
17     end.

```

- Giải thích:
 - Hàm laysochan(A) lấy những phần tử chia hết cho 2 bỏ vào list.
 - Hàm xuly(A) kiểm tra xem có số chẵn nào không, nếu không thì mảng toàn số lẻ, ngược lại ko toàn lẻ.
- Kết quả:



```
{ok, cau37}
34> cau37:xuly([12,18,24,1,1,2,3,4,69]).
Mang khong toan so le
ok
35> cau37:xuly([1,1,3,69]).
Mang toan so le
ok
36> cau37:xuly([0,1,3,69]).
Mang khong toan so le
ok
```

Câu 38.

- Code:

```
1 -module(cau38).
2 -export([tinh/1, tongAm/1]).
3
4 tinh([]) -> 0;
5 tinh([H|T]) ->
6     if H < 0 -> tinh(T) + H
7     ;true -> tinh(T)
8     end.
9
10 tongAm(N) ->
11     L = lists:append(N),
12     tinh(L).
```

- Giải thích:
 - Đưa ma trận N thành list L bằng hàm append.
 - Dùng hàm tính tính tổng các phần tử âm có trong L.
- Kết quả:

```
51> c(cau38).
{ok, cau38}
52> cau38:tongAm([[1,2,3],[2,-1,0]]).
-1
53> cau38:tongAm([[1,2],[2,-1],[-2,-5]]).
-8
54> cau38:tongAm([[1,2],[2,11],[3,5]]).
0
```

Câu 39

- Code:

```

1  |-module(cau39).
2  -export([list/1, kttoanle/2,listToanle/1,demToanle/1]).
3
4  list(0) -> [];
5  list(N) -> [N rem 10 | list(N div 10)].
6
7  kttoanle(_, 0) -> 0;
8  kttoanle([], 1) -> 1;
9  kttoanle([H|T], 1) ->
10     if (H rem 2 == 0) -> K = 0
11         ;true -> K = 1
12     end,
13     kttoanle(T, K).
14
15  listToanle([]) -> [];
16  listToanle([H|T]) ->
17     Ch = kttoanle(H,1),
18     if Ch == 1 -> [H] ++ listToanle(T)
19         ;true -> listToanle(T)
20     end.
21
22  demToanle(N) ->
23     L = lists:append(N),
24     length(listToanle(L)).

```

- Giải thích:
 - Lấy list các số toàn lẻ bằng hàm listToanle.
 - Đếm số phần tử của list này ta được kết quả.
- Kết quả:

```

64> c(cau39).
{ok,cau39}
65> cau39:demToanle([[1,2],[2,11],[3,5]]).
4
66> cau39:demToanle([[1,2],[2,11]]).
2
67> cau39:demToanle([[1,2],[2,-11]]).
2
---

```

Câu 40.

- Code:

```

1  -module(cau40).
2  -export([tim/3, nhieunhat/1]).
3
4  tim([], _, Val) -> Val;
5  tim(Lst, CurLen, CurVal) ->
6      [D|_] = Lst,
7      {Ht,C1} = lists:partition(fun(A) -> A == D end, Lst),
8      [H|_] = Ht,
9      NewLen = length(Ht),
10     if
11         NewLen > CurLen -> tim(C1, NewLen, H)
12         ;true           -> tim(C1, CurLen, CurVal)
13     end.
14
15 nhieunhat(N) ->
16     L=lists:append(N),
17     Max = tim(L,0,0),
18     io:fwrite("~p xuất hiện nhiều nhất~n", [Max]).

```

- Giải thích:
 - Hàm tim(Lst, CurLen, CurVal) lần lượt lấy hết các phần tử giống nhau trong list Lst, mỗi lần lấy so sánh số lượng với số lượng hiện tại, nếu nhỏ hơn thì lưu lại, cứ vậy đến khi hết list.
 - Hàm nhieunhat(L) chuyển ma trận thành mảng và sử dụng hàm tim tìm kiếm giá trị xuất hiện nhiều nhất và in kết quả.
- Kết quả:

```

Erlang
File Edit Options View Help
{ok,cau40}
45> cau40:nhieunhat([[1,1,1],[2,3,3]]).
1 xuất hiện nhiều nhất
ok
46> cau40:nhieunhat([[1,1,2],[2,3,3],[69,69,69]]).
69 xuất hiện nhiều nhất
ok
47> cau40:nhieunhat([[96,4],[2,3],[96,96]]).
96 xuất hiện nhiều nhất
ok
_

```

Câu 41.

- Code:

```

1 -module(cau41).
2 -export([transpose/1,sort_aRow/1,sortbyCol/1]).
3
4 transpose([]|_|_) -> [];
5 transpose(M) ->
6   [lists:map(fun hd/1, M) | transpose(lists:map(fun tl/1, M))].
7
8 sort_aRow([]) -> [];
9 sort_aRow([H|T]) ->
10   H1 = lists:sort(H),
11   [H1] ++ sort_aRow(T).
12
13 sortbyCol(M) ->
14   Lol = transpose(M),
15   transpose(sort_aRow(Lol)).
16

```

- Giải thích:
 - Biến dòng cột thành dòng, dòng thành cột trong ma trận L gán vào ma trận Lol bằng hàm transpose .
 - Sắp xếp các phần tử tăng dần theo hàm trong ma trận Lol.
 - Biến đổi cột thành dòng, dòng thành cột của Lol và trả về cho Hàm sortbyCol thì ta được kết quả cần làm.
- Kết quả:

```

70> c(cau41).
{ok,cau41}
71> cau41:sortbyCol([[5,6,9],[-2,4,3],[8,6,5]]).
[[-2,4,3],[5,6,5],[8,6,9]]
72> cau41:sortbyCol([[5,6,9],[-2,4,3]]).
[[-2,4,3],[5,6,9]]
73>

```

Câu 42.

- Code:

```

1 -module(cau42).
2 -export([trongdoan/3]).
3
4 trongdoan([],_,_) -> [];
5 trongdoan([H|T],X,Y) ->
6   {A,B,C} = H,
7   P = A + B + C,
8   if
9     P >= X, P <= Y -> [{A,B,C}] ++ trongdoan(T,X,Y)
10    ;true           -> trongdoan(T,X,Y)
11  end.

```

- Giải thích:
 - Hàm trongdoan(Lst, X, Y) tìm các tam giác có chu vi nằm trong đoạn [X, Y] thêm vào 1 list mới.
- Kết quả:

```
Erlang
File Edit Options View Help
in call from cau42:trongdoan/3 (cau42.erl, line 9)
20> c(cau42).
{ok,cau42}
21> cau42:trongdoan([3,4,5],[5,5,5],[6,7,9], 14, 22).
[5,5,5],[6,7,9]
22> cau42:trongdoan([3,4,5],[5,5,5],[6,7,9], 12, 20).
[3,4,5],[5,5,5]
23> cau42:trongdoan([3,4,5],[5,5,5],[6,7,9], 14, 15).
[5,5,5]
```

Câu 43.

- Code:

```
cau43.erl x helloworld.erl x
1 -module(cau43).
2 -export([xoa/2]).
3
4 xoa([], _) -> [];
5 xoa([H|T], X) ->
6     {A, B, C} = H,
7     P = A + B + C,
8     if
9         P >= X -> [{A, B, C}] ++ xoa(T, X)
10        ;true   -> xoa(T, X)
11    end.
```

- Giải thích:

- Hàm xoa(Lst, X) thêm các tam giác có chu vi $\geq X$ vào 1 list mới.

- Kết quả:

```
Erlang
File Edit Options View Help
cau43.erl:4: Warning: function xoa/2 is unused
error
27> c(cau43).
{ok,cau43}
28> cau43:xoa([3,4,5],[5,5,5],[6,7,9], 100).
[]
29> cau43:xoa([3,4,5],[5,5,5],[6,7,9], 1).
[3,4,5],[5,5,5],[6,7,9]
30> cau43:xoa([3,4,5],[5,5,5],[6,7,9], 15).
[5,5,5],[6,7,9]
```

Câu 44.

- Code:

```

1  -module(cau44).
2  -export([tangdan/3, kiemtratang/1]).
3
4  tangdan(_, 0, _) -> 0;
5  tangdan(_, _, 1) -> 1;
6  tangdan([H|[D|T]], State, Curlen) ->
7      {A, B, C} = H,
8      P1 = A + B + C,
9      {X, Y, Z} = D,
10     P2 = X + Y + Z,
11     if
12         P1 > P2 -> tangdan([D|T], 0, Curlen)
13         ;true   -> NewLen = length([D|T]), tangdan([D|T], State, NewLen)
14     end.
15
16 kiemtratang(A) ->
17     Curlen = length(A),
18     State = tangdan(A, 1, Curlen),
19     if
20         State == 1 -> io:fwrite("Chu vi tang dan~n")
21         ;true      -> io:fwrite("Chu vi khong tang dan~n")
22     end.

```

- Giải thích:
 - Hàm tangdan kiểm tra 1 list các tam giác có chu vi tăng dần hay không, nếu có trả về 1, ngược lại về 0.
 - Hàm kiemtratang in ra kết quả đã được kiểm tra.
- Kết quả:

```

Erlang
File Edit Options View Help
{ok,cau44}
52> cau44:kiemtratang([{3,4,5},{5,5,5},{6,7,9}]).
Chu vi tang dan
ok
53> cau44:kiemtratang([{3,4,5},{5,5,5},{2,2,2}]).
Chu vi khong tang dan
ok
54> cau44:kiemtratang([{6,6,9}]).
Chu vi tang dan
ok
_

```

Câu 45.

- Code:

```

1 | -module(cau45).
2 | -export([transpose/1, sort_aRow/1, sortbyCol/1]).
3
4 | transpose([]|_) -> [];
5 | transpose(M) ->
6 |   [lists:map(fun hd/1, M) | transpose(lists:map(fun tl/1, M))].
7
8 | sort_aRow([]) -> [];
9 | sort_aRow([H|T]) ->
10 |   H1 = lists:reverse(lists:sort(H)),
11 |   [H1] ++ sort_aRow(T).
12
13 | sortbyCol(M) ->
14 |   Lol = transpose(M),
15 |   transpose(sort_aRow(Lol)).

```

- Giải thích: Tương tự câu 41. Input vào các phân số nằm trong {}, vd: 7/9 input sẽ là {7,9}. Khác ở chỗ sắp xếp giảm dần.
- Kết quả:

```

76> c(cau45).
{ok,cau45}
77> cau45:sortbyCol([[{-2,7},{4,4},{3,5}],[{5,6},{6,8},{9,9}]]).
[[{5,6},{6,8},{9,9}],[{ -2,7},{4,4},{3,5}]]
78> █

```

Câu 46.

- Code:


```

1 |-module(cau46).
2 -export([chuvi/1,tSR/1,toLol/1,transpose/1,find/3,main/1]).
3
4 chuvi({H,T}) -> 2 * (H + T).
5
6 tSR([]) -> [];
7 tSR([H|T]) -> [chuvi(H)] ++ tSR(T).
8
9 toLol([]) -> [];
10 toLol([H|T]) -> [tSR(H)] ++ toLol(T).
11
12 transpose([[]|_]) -> [];
13 transpose(M) -> [lists:map(fun hd/1, M) | transpose(lists:map(fun tl/1, M))].
14
15 find(LL,N1,N2) ->
16     L = toLol(LL), % matrix chu vi HCN
17     Lt = transpose(L),
18     Cn2 = lists:nth(N2,Lt), % cot N2
19     SoDong = length(Cn2),
20     if N1 > SoDong -> []
21     ;true ->
22         Rn1 = lists:nth(N1,L), % dong N1
23         SoCot = length(Rn1),
24         K = lists:nth(N2,Rn1), % phan tu thu N2 cua dong thu N1 (ban dau)
25         MaxR = lists:max(Rn1),
26         if K == MaxR -> % neu max tren dong
27             Do1 = 0,
28             MinC = lists:min(Cn2),
29             if K == MinC -> Do2 = 0 % neu min tren cot -> stop
30             ;true -> Do2 = 1 % ko min -> continue
31             end
32         ;true -> Do1 = 1,Do2 = 1 % ko max -> continue
33         end,
34         if Do1 == 0,Do2 == 0 ->
35             Key = lists:nth(N2,lists:nth(N1,LL)), % thoa max tren dong, min tren cot -> in ra
36             if N2 < SoCot -> [Key] ++ find(LL,N1,N2+1)
37             ;true -> [Key] ++ find(LL,N1+1,1)
38             end
39         ;true ->
40             if N2 < SoCot -> find(LL,N1,N2+1) % chua het dong -> xet tiep tren dong
41             ;true -> find(LL,N1+1,N2) % het dong -> xet dong tiep theo
42             end
43         end
44     end.
45
46 main(N) -> find(N,1,1).

```

- Giải thích:
 - Hàm chuvi tính chu vi HCN với 2 cạnh co trước.
 - Biến ma trận HCN (gồm 2 cạnh nằm trong { }) thành ma trận chu vi HCN bằng hàm toLol, gán vào L.
 - Lấy list các phần tử của ma trận L thỏa điều kiện lớn nhất trên dòng và bé nhất trên cột bằng hàm find.
 - Hàm main trả về list kết quả.
 - Mô tả input và output:

MÔ TẢ TRỰC QUAN	
Input:	Output:
[[{0,1},{2,2}], {3,0},{2,3}]]	[2,8, 6,10] -> {2,2}
[[{3,3},{3,6},{6,3}], {2,1},{1,2},{6,3}]]	[12,18,18, 6, 6, 18] -> {6,3},{6,3}
[[{3,3},{6,0},{0,7},{2,5}], {0,3},{2,2},{5,2},{6,1}], {1,1},{2,6},{9,0},{5,3}]]	[12,12,14,14, 6, 8, 14,14, 4, 16,18,16] -> {0,7},{2,5},{5,2},{6,1}

- Kết quả:

```

78> c(cau46).
{ok,cau46}
79> cau46:main([[{0,1},{2,2}],[{3,0},{2,3}]]).
[{2,2}]
80> c(cau46).
{ok,cau46}
81> cau46:toList([[{0,1},{2,2}],[{3,0},{2,3}]]).
[[2,8],[6,10]]
82> cau46:main([[{0,1},{2,2}],[{3,0},{2,3}]]).
[{2,2}]
83> cau46:toList([[{3,3},{3,6},{6,3}],[{2,1},{1,2},{6,3}]]).
[[12,18,18],[6,6,18]]
84> cau46:main([[{3,3},{3,6},{6,3}],[{2,1},{1,2},{6,3}]]).
[{6,3},{6,3}]
85> cau46:toList([[{3,3},{6,0},{0,7},{2,5}],[{0,3},{2,2},{5,2},{6,1}],[{1,1},{2,6},{9,0},{5,3}]]).
[[12,12,14,14],[6,8,14,14],[4,16,18,16]]
86> cau46:main([[{3,3},{6,0},{0,7},{2,5}],[{0,3},{2,2},{5,2},{6,1}],[{1,1},{2,6},{9,0},{5,3}]]).
[{0,7},{2,5},{5,2},{6,1}]
87> █

```

Câu 47.

- Code:

```

1  -module(cau47).
2  -export([tangdan/3, kiemtratang/2, kiemtra/1]).
3
4  tangdan(_, 0, _) -> 0;
5  tangdan(_, _, 1) -> 1;
6  tangdan([H|[D|T]], State, CurLen) ->
7      {A, B, C} = H,
8      P1 = A + B + C,
9      {X, Y, Z} = D,
10     P2 = X + Y + Z,
11     if
12         P1 > P2 -> tangdan([D|T], 0, CurLen)
13         ;true  -> NewLen = length([D|T]), tangdan([D|T], State, NewLen)
14     end.
15
16 kiemtratang(_, 1) -> 1;
17 kiemtratang([], 0) -> 0;
18 kiemtratang([H|T], State) ->
19     CurLen = length(H),
20     Status = tangdan(H, 1, CurLen),
21     if
22         Status == 1 -> kiemtratang(T, 1)
23         ;true       -> kiemtratang(T, State)
24     end.
25
26 kiemtra(A) ->
27     Status = kiemtratang(A, 0),
28     if
29         Status == 1 ->
30             io:fwrite("Ma tran co ton tai dong co chu vi tang dan~n")
31         ;true -> io:fwrite("Ma tran khong ton tai dong co chu vi tang dan~n")
32     end.

```

- Giải thích:
 - Hàm tangdan kiểm tra 1 list các tam giác có chu vi tăng dần hay không, nếu có trả về 1, ngược lại về 0.
 - Hàm kiemtratang kiểm tra từng dòng của ma trận, nếu có dòng có chu vi tăng trả về 1, ngược lại về 0.
 - Hàm kiemtra kiểm tra và in kết quả.
- Kết quả:

```

Erlang
File Edit Options View Help
{ok,cau47}
22> cau47:kiemtra([[{3,4,5},{5,5,5}],[{6,5,9},{4,7,5}]]).
Ma tran co ton tai dong co chu vi tang dan
ok
23> cau47:kiemtra([[{6,6,5},{5,5,5}],[{6,5,9},{4,7,5}]]).
Ma tran khong ton tai dong co chu vi tang dan
ok
24> cau47:kiemtra([[{6,6,5},{5,5,5},{35,69,96}],[{6,5,9},{4,7,5},{1,1,1}]]).
Ma tran khong ton tai dong co chu vi tang dan
ok

```

Câu 48.

- Code:

```

50 Diem.
51
52 layHT([]) -> [];
53 %layHT(_, 0) -> [];
54 layHT([H|T]) ->
55     [layhoten(H) | layHT(T)].
56
57 top3([], _) -> "";
58 top3(_, 0) -> "";
59 top3([H|T], I) ->
60     io:fwrite("~p~n", [H]),
61     top3(T, I - 1).
62
63 main() ->
64     {ok, File} = file:open("input.txt", [read]),
65     [_|DanhSach] = read(File),
66     file:close(File),
67     DanhSach,
68     DShoten = layHT(DanhSach),
69     io:fwrite("Cau a:~n"),
70     io:fwrite("~p~n", [lists:sort(fun(A, B) -> layten(A) < layten(B) end, DShoten)]),
71     DSten = layten(DanhSach, 1),
72     Mapten = maps:from_list(DSten),
73     io:fwrite("Cau b:~n"),
74     io:fwrite("~p~n", [maps:keys(Mapten)]),
75     io:fwrite("Cau c:~n"),
76     top3(lists:sort(fun(A, B) -> laydiem(A) > laydiem(B) end, DanhSach), 3).

```

- Giải thích:
 - Câu a: dùng hàm sort trong module lists để sắp xếp.
 - Câu b: dùng map để lưu với key là các tên, sau đó in ra các key.
 - Câu c: sắp xếp danh sách theo điểm giảm dần rồi in ra 3 người đầu tiên.
- Kết quả:

```

1 6
2 1712003, Vu Anh Quan, 8.9
3 1712033, Nguyen Van Chung, 8.7
4 1712033, Hoang Trong Quynh, 9.1
5 1712038, Dinh quang Huy, 8.5
6 1712039, Dinh Quang Hoang, 8.8
7 1712044, Cong Hoang, 9.0

```

```

21> c(cau48).
{ok,cau48}
22> cau48:main().
Cau a:
["Nguyen Van Chung","Cong Hoang","Dinh Quang Hoang","Dinh quang Huy",
 "Vu Anh Quan","Hoang Trong Quynh"]
Cau b:
["Chung","Hoang","Huy","Quan","Quynh"]
Cau c:
["1712033","Hoang Trong Quynh","9.1"]
["1712044","Cong Hoang","9.0"]
["1712003","Vu Anh Quan","8.9"]
[]

```

Câu 49.

- Code:

```

khachhang.txt x giaithuong.txt x cau49.erl x
37 laysogiai([_|Sogiai]) ->
38 [X|_] = Sogiai,
39 list_to_integer(X).
40
41 xuấtDS([], _) -> [];
42 xuấtDS(L, 0) -> L;
43 xuấtDS([H|T], N) ->
44 io:fwrite("~p, ", [layma(H)]),
45 xuấtDS(T, N - 1).
46
47 xuất(_, []) -> "";
48 xuất([], _) -> "";
49 xuất(Danhsach, [Hgiai|Tgiai]) ->
50 io:fwrite("~p: ", [laytengiai(Hgiai)]),
51 DS = xuấtDS(Danhsach, laysogiai(Hgiai)),
52 io:fwrite("~n"),
53 xuất(DS, Tgiai).
54
55 main() ->
56 {ok, File} = file:open("khachhang.txt", [read]),
57 [_|Danhsach] = read(File),
58 file:close(File),
59 Danhsachsx = lists:sort(fun(A, B) -> laydiem(A) > laydiem(B) end, Danhsach),
60 {ok, FileGiaiThuong} = file:open("giaithuong.txt", [read]),
61 [_|GiaiThuong] = read(FileGiaiThuong),
62 file:close(FileGiaiThuong),
63 GiaiThuongsx = lists:sort(fun(A, B) -> laytengiai(A) < laytengiai(B) end,
64 GiaiThuongsx).

```

- Giải thích:
 - Đọc danh sách khách hàng và sắp xếp giảm dần theo điểm tích lũy.
 - Đọc danh sách giải và sắp xếp tăng dần theo tên giải.
 - Sau đó xuất danh sách khách hàng theo số lượng giải của mỗi giải.
- Kết quả:

khachhang.txt	giaithuong.txt
1 6	1 3
2 KH17120, Vu Anh Quan, 1991, 1000	2 Giai 2, 2
3 KH17121, Nguyen Van Chung, 1978, 2000	3 Giai 1, 1
4 KH17123, Hoang Trong Quynh, 1987, 2400	
5 KH17138, Dinh quang Huy, 2000, 300	
6 KH17139, Dinh Quang Hoang, 1968, 3000	
7 KH17144, Cong Hoang, 1995, 1400	

```

25> c(cau49).
{ok,cau49}
26> cau49:main().
"Giai 1": "KH17139",
"Giai 2": "KH17123", "KH17121",
[] _

```

Câu 50.

- Code:

```

1 -module(cau50).
2 -import(rand,[uniform/1]).
3 -export([sotrungthuong/0]).
4
5 ▼ sotrungthuong() ->
6     S1 = uniform(10) - 1,
7     S2 = uniform(10) - 1,
8     S3 = uniform(10) - 1,
9     S4 = uniform(10) - 1,
10    S5 = uniform(10) - 1,
11    S6 = uniform(10) - 1,
12    io:fwrite("~p~p~p~p~p~p~n", [S1, S2, S3, S4, S5, S6]).|

```

- Giải thích:
 - Hàm sotrungthuong sinh ngẫu nhiên 1 số trúng thưởng xổ số kiến thiết.
- Kết quả:

```

Erlang
File Edit Options View Help
66> cau50:sotrungthuong().
269430
ok
67> cau50:sotrungthuong().
346850
ok
68> cau50:sotrungthuong().
998831
ok

```

Câu 51.

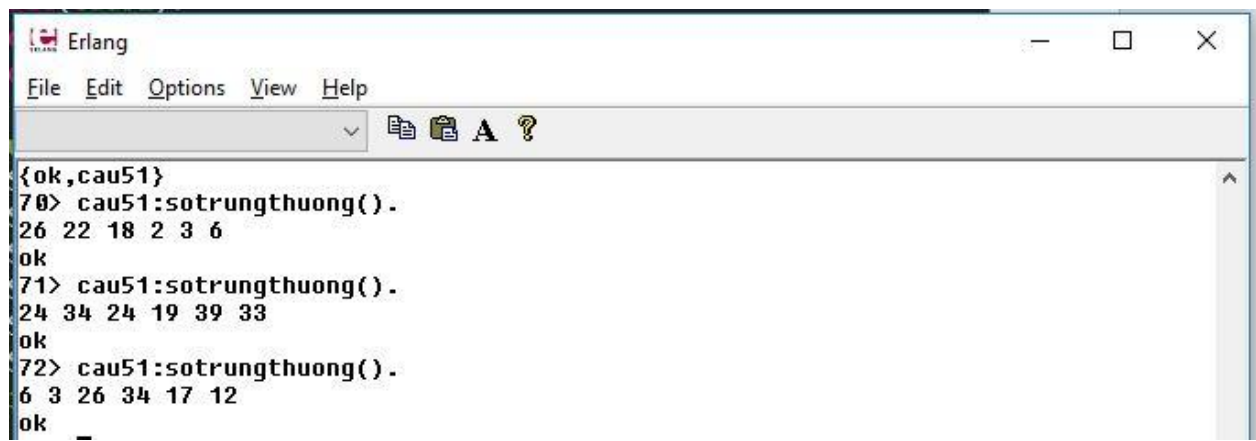
- Code:

```

1 -module(cau51).
2 -import(rand,[uniform/1]).
3 -export([sotrungthuong/0]).
4
5 sotrungthuong() ->
6     S1 = uniform(45),
7     S2 = uniform(45),
8     S3 = uniform(45),
9     S4 = uniform(45),
10    S5 = uniform(45),
11    S6 = uniform(45),
12    io:fwrite("~p ~p ~p ~p ~p ~p ~n", [S1, S2, S3, S4, S5, S6]).

```

- Giải thích:
 - Hàm sotrungthuong sinh ngẫu nhiên 1 số trúng thưởng xổ số Vietlott.
- Kết quả:



```
{ok,cau51}
70> cau51:sotrungthuong().
26 22 18 2 3 6
ok
71> cau51:sotrungthuong().
24 34 24 19 39 33
ok
72> cau51:sotrungthuong().
6 3 26 34 17 12
ok
_
```