1. Wybierz prawdziwe wyrażenie (Ada)
a) (2+3)*4 = 2+(3*4)
b) 6/4 = 1.5
c) 6/2 > 21 – 1
d) 2+8 /= 28

2. Która deklaracja procedury jest nieprawidłowa
a) procedure Delete_File(Integer : in A);
b) procedure Delete_File(X : in Integer);
c) procedure Delete_File(FOR : in A); - for jest słowem zarezerwowanym
d) procedure Delete_File(Y : in Float);

3. Operator przypisania w j. Ada można przeładować:
a) nie można tego zrobić
b) tylko w typie pochodnym
c) można, ale tylko w wersji Ada95
d) tylko w wersji komercyjnej kompilatora

4. Dla typu ograniczonego w Adzie (wybierz prawdziwe zdanie):
a) nie można zdefiniować operatorów :=, =, /=
b) wartości są ograniczone tylko do podanego podzakresu
c) nie są automatycznie generowane operatory :=, =, /=
d) operatory :=, =, /= automatycznie trafiają do sekcji prywatnej

5. Parametr klasowy T'Class w Adzie (zakreśl fałsz)
a) ma zawsze stały rozmiar i dlatego pozwala na wskazywanie dowolnego typu pochodnego
b) obejmuje wszystkie typy wyprowadzone z T
c) pozwala na przechowywanie wartości dowolnego typu wyprowadzonego z T
d) wchodząc w skład parametrów podprogramów wyklucza je z operacji podstawowych

6. Zakreśl poprawne zdanie odnoszące się do wykonania w przeplocie:
a) nowe zadanie po zakończeniu poprzedniego
b) kilka zadań wykonywanych jednocześnie
c) jedno zadanie wykonywane w danym momencie, kilka rozpoczętych
d) następne zadanie rozpoczynane tylko po zakończeniu poprzedniego

7. CCS dopuszcza, aby definicje agentów były wzajemnie rekurencyjne
(wybierz prawdę)
a) tylko dla agentów niewykorzystujących później rekurencji
b) tylko dla agenta pustego
c) tylko jeśli zostanie użyty operator złożenia równoległego
d) zawsze – nie znalazłem żadnych obostrzeń

8. Zaznacz odpowiedź zawierającą tylko poprawne identyfikatory j. Ada.
a) Hello, 2Run, QuitNow – musi się zaczynać od litery
b) Quit Now, RemoveElEM – nie może zawierać spacji
c) Add_Delta, XXX
d) 2-gi_Rocznik, TestQuest – nie może zaczynać się od cyfry, ani zawierać znaku specjalnego innego niż _

9. Typy kontrolowane w Adzie (wybierz prawdę)
a) wymuszają zdefiniowanie metod wykonywanych przy: inicjalizacji, finalizacji i poprawianiu po przypisaniu
b) muszą być zdefiniowane dla typów znakowych, bo inaczej będą one źle działać
c) pozwalają programiście na zdefiniowanie metod wykonywanych przy: inicjalizacji, finalizacji i poprawianiu przy przypisaniu
d) muszą być zdefiniowane dla typów znakowych (pochodnych Character) Przez eliminację ;-)

10. lists:zipwith(fun(X,Y) -> {X, Y} end,[1,2,3],[a,b,c]). Wynikiem będzie:
a) [{1,a},{2,b},{3,c}]
b) [{a,b,c},{1,2,3}]
c) [{1,a},{b,2},{3,c}]
d) {{1,a},{2,b},{3,c}}

11. Który z elementów nie jest atomem dla Erlanga Atom zaczyna się małą literą lub jest wzięty w znaki ' '
a) a
b) Elka
c) 'Elka'
d) zEtka

12. Bariery nie są wartościowane w obiekcie chronionym
a) po wykonaniu procedury operującej na wartościach argumentów będących elementami bariery
b) po wykonaniu funkcji
c) po opuszczeniu sekcji krytycznej przez zadanie będące wcześniej w kolejce wywołań
d) przy wywołaniu wejścia

13. <<A:4,B:4,C:8>> = <<56,66>>.
a) A=5,B=6,C=66
b) A=3,B=8,C=8#102
c) A=4,B=6,C=2#001000010
c) A=1,B=6,C=65
<<55,66>> w zapisie bitowym: 0011 1000 | 0100 0010, czyli A = 3, B = 8, C = 102 w zapisie 8-kowym, czyli 66.

<<A:3, B:4, C:8>> == <<56, 66>>
a) obie wymienione wartości są prawidłowe
b) A = 3, B = 8, C = 8#102
c) żadne z wymienionych
d) A = 3, B = 8, C = 66

14. Przy deklaracji: type Day is (Mon, Tue, Wed, Thu, Fri, Sat, Sun); wartość wyrażenia Day'Pred(Thu) to:
a) 2
b) Wed
c) Day
d) Mon

15. Wskaż niepoprawną deklarację:
a) procedure DFile(Int: in A)
b) function DFile(X: in out A) return D //funkcja musi mieć wszystkie parametry typu in
c) procedure DFile(X: out Integer)
d) function DFile(Y: in A) return D

16. lists zipwith(fun(X,Y) - > [X,[X|[Y]]] end, [1,2,3] , [a,b,c]) wynik:
a)[[1,[1,a]],[2,[2,b]],[3,[3,c]]]
b)[[1,[1|a]],[2,[2|b]],[3,[3|c]]]
c)[{1,1,a},{2,2,b},{3,3,c}
d){[1,a,1],[2,b,2],[3,c,3]}

17. lists:foldl(fun(X, Sum) -> X – Sum end, 0, [1,2,3,4,5]). Wynikiem będzie:
a)-13
b)3
c)15
d)5

18. Co wypluje wyrażenie: "lists:foldl(fun(X,Sum) -> X - Sum end, 3, [X-1 || X <- lists:seq(1,5, X > 2])."
a) wtf
b) 0
c) 3
d) ...

19. Instrukcja abort powoduje (zaznacz prawdę)
a) natychmiastowe, bezwarunkowe zakończenie zadania
b) usunięcie zadania z kolejki oczekujących na wejście do ob. chronionego
c) ustawienie statusu Va_t'em
d) wywołanie procedury stop() zadania

20. [...] w obiekcie chronionym (zakreśl prawidłową)
a) po każdej zmianie kontekstu
------b) po wykonaniu funkcji o. ch.
c) po każdym wznowieniu zadania o wyższym priorytecie niż to, które będzie wykonywać składowe o. ch.
d) przy wywołaniu wejścia

Różnice pomiędzy funkcją a procedurą w Adzie:

Funkcja:	Procedura
przeciążanie tryb in wywołanie jako część wyrażenia	przeciążanie domyślny tryb in można wstawić instrukcję return przykrywanie zmiennych

---

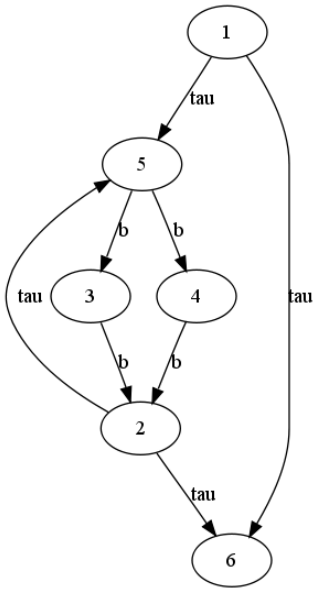
Napisz fragment kodu w języku Ada który pozwoli max przez 5 sek. Oczekiwać na wywołanie wejścia innego zadania. Przy niepowodzeniu ma być wywoływane wejście zadania „zapasowego”.

```
select
  A.E;
or
  delay 5.0;
  B.E;
end select;
```

Napisz w Adzie zadanie generujący impuls o zadanej długości. Zadanie ma posiadać wejście do wyzwalania impulsu i precyzowania jego długości. Impuls generowany jest przez wywołanie procesów ob. chronionego **Sig: On i Off**

```
task type Sig
  entry LOL(T: in Duration);
end Sig;

task body Signal is
Time: Duration;
begin
  select
    accept LOL(T);
    Time = T;
  end select
  SIG.On();
  delay T;
  SIG.Off();
end Sig;
```

CCS: narysować graf stanów dla:
A=a.b.A + c.0
B='a.b.B + 'c.B
C=(A|B)\{a,c}



Napisz kod w języku Erlang który będzie implementacją następującej maszyny stanowej:
A(a)->A, A(b)->B, A(c)->C,B(f)->F, B(c)->C, F(f)->A, C(f)->D, D(g)->A, D(f)->F

```
-module(stan)
-export([start\0], [masz\1], change[\1])

start() ->
Pid = spawn(stan, masz, 'A'),
register(machine, Pid)
end.

change(X) ->
Pid = wheris(machine),
Pid ! X
end.

masz('A') ->
receive
   a -> masz('A');
   b -> masz('B');
   c -> masz('C');
   d -> masz('d');
   _ -> masz('A')
end.

masz('B') ->
receive
   c -> masz('C');
   f -> masz('F');
   _ -> masz('B')
end.

masz('C') ->
receive
   g -> masz('G');
   _ -> masz('C')
end.

masz('F') ->
receive
   p -> masz('A');
   _ -> masz('F')
end.

masz('G') ->
receive
   w -> masz('A');
   _ -> masz('G')
end.

masz('D') ->
receive
   a -> masz('A');
   f -> masz('F');
   _ -> masz('D')
end.
```

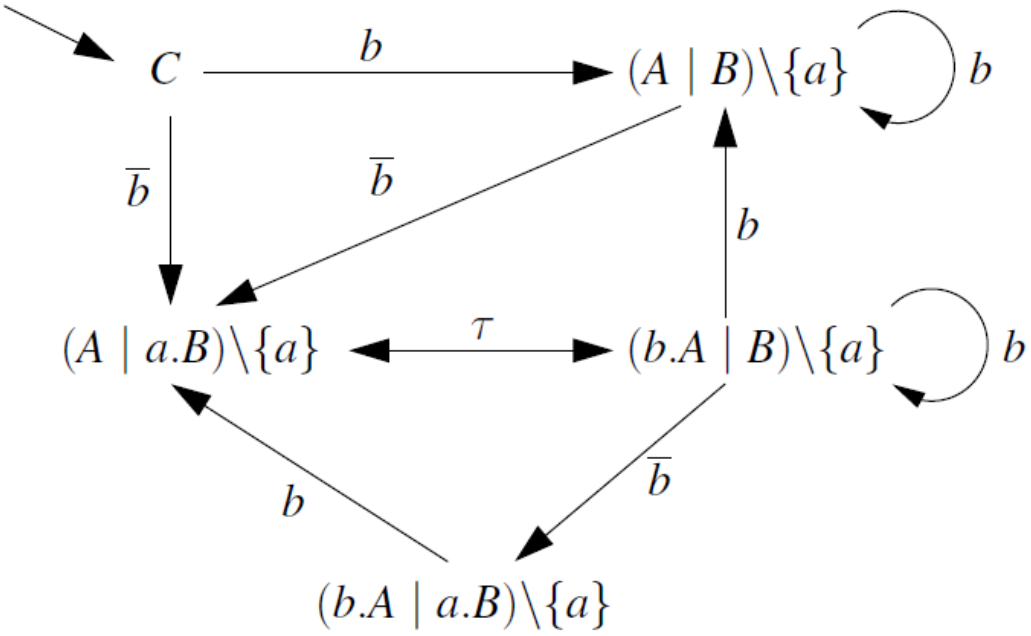
Napisz w Adzie szkielet programu do obsługi przerwań (sygnałów).

```
protected Sensor is
     procedure Int_Handler;
     pragma Attach_Handler(Int_Handler, InterruptID);
private
     -- ...
end;
protected body Sensor is
     procedure Int_Handler is
     begin
          -- ...
     end;
     -- ...
end;
```

Napisz fragment kodu w języku Ada, który pozwoli max przez 5sek oczekiwać na zakończenie wykonania procedury „FixWorld(W:World)". Jeśli procedura nie zakończy się w tym czasie, to ma zostać wyświetlony komunikat.

```
with Ada.Text_IO;
use Ada.Text_IO;

procedure zad16 is

type World is range 1..10;
W:World := 5;

procedure FixWorld(S:World) is
begin
     Put_Line("World fixed!");
end;

task A is
     entry E;
end;

task body A is
begin
     delay 10.0; -- opoznienie, zeby A.E nie moglo sie wykonac
     accept E;
     FixWorld(W);
end;

begin
     select
          A.E;
     or
          delay 5.0;
          Put_Line("World cannot be fixed!");
     end select;
end zad16;
```

Narysuj graf stanów osiągalnych dla następującego systemu zapisanego w CCS
A = 'a.b.A + a.b.0
B = 'b.a.B + b.B
C = (A | B) \ {a}



Napisz kod w języku Erlang, który uruchomi 2 procesy A i B. Proces A będzie wysyłał do B wiadomość co 1sek i oczekiwał max 200ms na odpowiedź. Jeśli odpowiedź nie nadejdzie, to wyświetli komunikat i zakończy działanie.

```
-module(procesy).
-export([procA/1, procB/0]).

procA(Pid) ->
     Pid ! {msg,self()},
     receive
          ans ->
               io:format("Proces A odebral odpowiedz.~n",[]),
               timer:sleep(1000),
               procA(Pid)
     after 200 ->
          io:format("Proces A nie odebral odpowiedzi.~n",[])
     end.

procB() ->
     receive
          {msg,Pid} ->
               io:format("Proces B odebral wiadomosc.~n",[]),
               Pid ! ans
     end.

Pid = spawn(procesy, procB, []),
spawn(procesy, procA, [Pid]).
```

Napisz w Adzie i erlangu program tworzący 100 procesów, które czekają na wiadomość, po której kończą działanie

ADA

```
with Ada.Text_IO;
use Ada.Text_IO;

procedure zad1 is


task type zadanie is
    entry die;
end;

task body zadanie is
begin
    accept die;
end;

zadania: array (1..100) of zadanie;

task start;
task body start is
begin
    for i in 1..100
    loop
        zadania(i).die;
    end loop;
end;

begin
    null;
end zad1;
```

ERLANG

```
-module(zad1_raw).
-export([start/0]).

start() ->
    L = for(1, 100, fun() -> spawn(fun() -> wait() end) end),
    lists:foreach(fun(Pid) -> Pid ! die end, L).

wait() ->
    receive
        die ->
            void
end.

for(N,N,F) -> [F()];
for(I,N,F) -> [F()|for(I+1,N,F)].
```

szkielet obsługi wyjątków w Adzie

```
begin
-- . . .
exception
when Constraint_Error | Program_Error =>
. . .
when Storage_Error
. . .
when others =>
. . .
end ;
```

Erlang - napisać program, w którym funkcja startowa odpali 100 procesów, następnie wyśle do pierwszego odpalonego komunikat i sama zaczeka na komunikat. Każdy proces po otrzymaniu komunikatu ma wysłać komunikat do następnego procesu. Ostatni proces wysyła komunikat do funkcji startowej, po czym program się kończy.

```
-module(kol5).
-export([s/1, p/0, c/1]).

p() ->
   receive
      {PrevPID, [NextPID|Rest]} ->
         io:format("my:~w from:~w to:~w~n", [self(), PrevPID, NextPID])
   end,
   NextPID ! {self(), Rest}.

c(0) ->
   [self()];

c(N) ->
   [spawn(kol5, p, [])|c(N - 1)].

s(N) ->
   [FirstPID|Rest] = c(N),
   io:format("my:~w to:~w~n", [self(), FirstPID]),
   FirstPID ! {self(), Rest},
   receive
      {PrevPID, []} ->
         io:format("my:~w from:~w~n", [self(), PrevPID])
end.
```

napisać w erlangu program tworzący 100 procesów, które kończą swój żywot po otrzymaniu wiadomości "die"

```
-module (zad2).
-export ([start/0, zabij/1, for/2, cos/0]).
start () ->
    Pid_List = for (100, []),
    zabij (Pid_List).
zabij ([]) ->
    io:format ("Wszystkie martwe~n", []);
zabij ([H | R]) ->
    H ! die,
    io:format ("Zabij~n", []),
    zabij (R).
for (X, L) when X == 0 ->
    L;
for (X, L) ->
    for(X-1, [spawn (zad2, cos, []) | L]).
cos () ->|
    receive
        die ->
            io:format ("Martwy ~w~n", [])
end.
```

TO SAMO W ADZIE

```
with Ada.Text_IO,Ada.Integer_Text_IO;
use Ada.Text_IO;
procedure zad1 is
task type proces is
    entry die;
end;
task body proces is
begin
    accept die;
    put_line ("umieram");
end;
procesy: array (Integer range 1..100) of proces;
begin
    for i in 1..100
    loop
        procesy(i).die;
    end loop;
end zad1;
```

Skutki uboczne programowania.
Jeżeli fragment kodu zmienia stan zmiennej/pliku/czegokolwiek leżącej poza tym fragmentem (lub syntactic scope) to mówimy o skutkach ubocznych.

Bezpieczenstwo systemów współbieżnych - odpowiedz by ptm "nic złego sie nie stanie"
Co do programów współbieżnych, to cechują je bezpieczeństwo i żywotność. Pierwsze oznacza, że nic złego nigdy się nie wydarzy, a drugie, że kiedyś tam w końcu uzyskamy zamierzony cel (tj. na przykład każdy filozof sobie zje, każdy dostanie w końcu 2 widelce).

czym sie rożni przeplot od współbieżności
- Przeplot - raz jeden proces, raz drugi
- współbieżność - np. dwa procesy na różnych procesorach

krotka a lista w erl
- Krotka - zawiera się w nawiasach klamrowych '{}', nie może być rozszerzana, długość i kolejność argumentów jest istotna
- Lista - zawiera się w nawiasach [], można łatwo rozszerzać (przez A=[a]. B=[b,A]), oraz dopasowywać (G=[a,b,c,d,e,f].[O|_]=G.)

czy można przypisać X:=Y gdzie x i y to obiekty typu array of integer 1..10
nie, nie można bezpośrednio tablicy tak podstawić

jak działa kod : select cos or select cos or terminate (ada)
Select jeden opcji, jeśli nie to druga, jeśli żadna z powyższych to zabij.

czy x++; w c jest atomowy
Nie, MOV i INC.

inwersja priorytetów – wyjaśnij
inwersja priorytetów to zdobycie przez zadanie będące w sekcji krytycznej pierwszeństwa w dostępie do procesora kosztem zadań o wyższych priorytetach

przypisz do FUN1 funkcje która zwraca podwojony argument
FUN1 = fun (X) -> 2*X end.

jaki da wynik funkcja w erlangu -> zwykły map {x,2x} dla [1,2,3,4]
[{1,2},{2,4},{3,6},{4,8}]

Ada - zdefiniuj nowy typ - elementy A B C D
type Aligator is (A,B,C,D)

Czy wskaźniki w Adzie implementowane są tak samo jak wskaźniki w C/C++? Uzasadnij.
Tak. Używana jest bardzo podobna składnia przy implementowaniu jednych jak i drugich.

Scharakteryzuj typ abstrakcyjny w Adzie.
Typ znakowany można zdefiniować jako typ abstrakcyjny. Wówczas niektóre z metod można zdefiniować jako abstrakcyjne. Nie można tworzyć obiektów typu abstrakcyjnego. Klasę abstrakcyjną można wyprowadzić z innej klasy abstrakcyjnej lub z klasy konkretnej. W każdym przypadku odziedziczone konkretne metody można zastąpić metodami konkretnymi lub abstrakcyjnymi, można dodać nowe metody (konkretne lub abstrakcyjne) itd.

Na czym polega zagłodzenie procesu?
w informatyce sytuacja w środowisku wielozadaniowym, w której dany proces nie jest w stanie zakończyć działania, ponieważ nie ma dostępu do procesora lub innego współdzielonego zasobu.

lists:map(fun(X) -> [X], [1,2,3,4,5]). - jaki będzie wynik?
[[1], [2], [3], [4], [5]]

co oznacza "sprawiedliwość" w systemach współbieżnych
Każdy proces dostaje swój "czas antenowy"

Object(C):=Object(T) - co zrobi ten kod
Rzutowanie C i T na Object, a następnie przypisanie T do C.

napisać w erlangu funkcję pobierającą 2 argumenty i zwracającą ich sumę
fun (X,Y) -> X+Y end.

lists:zipwith(fun(X)->{X}, [a,b,c,d]) - jaki będzie wynik
[{a}, {b}, {c}, {d}]

czy instrukcja "spawn_link" w erlangu jest atomowa/why/why not ?
Nie jest - to złożenie spawn i link.

3 sposoby szeregowania zadań
sprawiedliwość, priorytetowy, fifo