CO545 Spring Term 2020 Assessment 2 SOLUTION

1. This question takes the example of fundraising through crowdfunding.

   Erlang Enterprises (EE) are trying to raise a sum of money to support their further development. They ask for bids from people, which have the form `{atom, int}`. For example joe bidding £1000 is represented by the pair `{joe,1000}`.

   All the bids (in the order in which they arrived, earliest first) are given in a list, e.g.

   ```
   [ {joe,1000}, {robert,3000}, {grace,5000}, {ada, 500} ]
   ```

   We call this list the *bid list.*

   Note: In answering this question, you may use any Erlang library function in your solution, as long as you explain clearly what it does in general.

   (a) Bids should not be zero or negative. Write a function `pos_bids` which takes a bid list `Bids` and returns a list leaving out any bids that do not contain a positive number as the bid amount. (You may assume that all the bids do contain a number in the second place, and so you don't have to check for that.) [6 marks]

```
pos_bids(Bids) -> lists:filter(fun ({_,A}) -> A>0 end, Bids).
```

   The solutions is illustrative only; other approaches are equally acceptable. For each part award full marks for an essentially correct solution, ignoring minor syntactic infelicities. Award half marks for solutions that show correct insights into the solution, and higher/lower according to the extent and insight of the solution.

   The same remarks apply to (b) and (c) too.

   (b) Suppose that EE have set a threshold for the crowdfunding exercise, and that if the total of the bids is less than this amount then all should be returned to the bidders. Write a function `success` that takes `Bids` and `Threshold` as arguments and checks whether the sum of all the bid amounts in the list `Bids` is at least the value given in the `Threshold` parameter. [6 marks]

```
sum_bids(Bids) -> lists:sum(lists:map( fun ({_,A}) -> A end, Bids)).

success(Threshold,Bids) -> sum_bids(Bids) >= Threshold.
```

   (c) Suppose that the list `Bids` does contain enough to exceed the `Threshold`, as checked in (b). Write a function `winners` that takes `Bids` and `Threshold` as arguments, and returns the list of bids that have been successful. These will be taken from the front of the list until the `Threshold` is exceeded.

   For example, if `Bids` is the list given at the start of the question, then the result of `winners(5000, Bids)` will be the list

```
[ {joe,1000}, {robert,3000}, {grace,1000} ]
```

Note here that `grace`'s bid has been lowered to make the total precisely the threshold value, which is `5000` here. [8 marks]

```
winners(_Threshold,[]) -> should_not_happen;
winners(Threshold,[{N,A}|Bids]) when Threshold>A ->
        [{N,A} | winners(Threshold-A,Bids)];
winners(Threshold,[{N,A}|_Bids]) when Threshold=<A -> [{N,Threshold}].
```

2. This question concerns strings, like `"foo"` and `"football"`.

(a) Define a function `init` in Erlang that takes two strings and returns true if the first string is an initial segment of the other: in other words, if the first string can be extended to make the second. For example,

`init("foo","football")` should be `true`, and

`init("foo","ballfoot")` should be `false`.

The solution has the form as follows. 1+1+2 marks for the three clauses.

```
init([],_) -> true;
init(_,[]) -> false;
init([X|Xs],[Y|Ys]) ->
    X==Y andalso init(Xs,Ys).
```

[4 marks]

(b) Define a function drop that takes an integer `N` and a string `St`, and which returns the string `St` with the first `N` elements dropped, if the string has that many elements (and as many as possible otherwise). For example,

`drop(2,"football")` should be `"otball"`, and

`drop(12,"football")` should be `""`.

The solution has the form as follows. 1+1+2 marks for the three clauses.

```
drop(0,Xs) -> Xs;
drop(_,[]) -> [];
drop(N,[_|Xs]) -> drop(N-1,Xs).
```

[4 marks]

(c) Using the functions `init` and `drop`, or otherwise, define a function `subst` that takes three strings, `Old`, `New` and `St`. The function returns a string in which the first occurrence of `Old` is replaced by `New`; if it doesn't occur, then the string is returned unchanged. For example:

`subst("foo","bar","football")` should be `"bartball"`, and

`subst("foo","bar","ballfoot")` should be `"ballbart"`, and

`subst("foo","bar","footfoot")` should be `"bartfoot"`. [8 marks]

The solution has the form as follows. 8 marks for a broadly correct solution, modulo some syntactic errors, 6 marks for a solution that is close to correctness, and 4 marks for a solution that shows some germ of a solution .

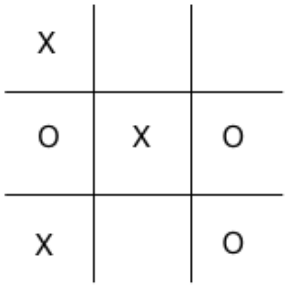
```
subst(Old,New,St) ->
    case init(Old,St) of
        true  -> New ++ drop(length(Old),St);
        false -> case St of
                     [X|Xs] -> [X|subst(Old,New,Xs)];
                     [] -> St
                 end
    end.
```

(d) How would you modify your answer to (c) so that *all* occurrences of `Old` are replaced by `New`? How would you modify it so that only the *last* occurrence of `Old` was replaced?

- Need to make a recursive call to `subst` in the true case of the outer pattern match (or as appropriate for the student's particular solution).
- One way would be to reverse the two strings that are arguments to the original function, and then to reverse the result.

These could be described in words (in a comment) or given as actual code. [2+2 marks]

3. We can represent the state of a noughts and crosses board by a list of three lists, one for each line of the board. For instance, the board:



is represented by `[[x,b,b],[o,x,o],[x,b,o]]` where `x`, `b`, `o` are atoms (and `b` stands for "blank").

(a) Define a function `isxwin` in Erlang that will take a line of a board and return a Boolean saying whether or not the line is a winning line of crosses, i.e. it consists of three crosses. [2 marks]

```
isxwin([x,x,x]) ->

    true;


isxwin(_) ->

     false.
```

2 marks: one per part. Be lenient with minor syntactic deviations.

(b) Using your solution to (a) or otherwise, define a function in Erlang that takes a board and returns a Boolean if the board contains a winning line (horizontal only) of crosses. [4 marks]

```
linewin(Xs) ->

     lists:foldl(fun(X,Y) -> X orelse Y end, false,

          lists:map(fun isxwin/1,Xs)).
```

4 marks: Numerous solutions. Fine to match a three element list too. Be lenient with minor syntactic deviations.

(c) Define a function `pick` in Erlang that takes an integer `N` and a list `Xs` and returns the `N`th element of the list, starting counting from `0`. For example:

```
pick(0,[a,b,c]) = a

pick(2,[a,b,c]) = c
```

You can assume that the function is called with a value of `N` that makes sense for `Xs` (in the case of the list `[a,b,c]` it is `0`, `1` or `2`). [6 marks]

```
pick(0,[X|_]) ->

       X;

pick(N,[_|Xs]) when N>0 ->

     pick(N-1,Xs).
```

6 marks: Numerous solutions. 2 marks for the base case and 4 for the recursion. Be lenient with minor syntactic deviations. No penalty for defensive programming. The solution could also be written using lists:nth, e.g.:

```
pick(N,Xs) -> lists:nth(N+1,Xs).
```

But this **must** use nth in the right way.

(d) Using your answer to (c) or otherwise, define a function `wincol` in Erlang that takes a board and returns `True` if the board contains a winning column of crosses. For example, the board shown at the start of the question does not contain a winning, but the board:

```
[[x,o,b],[o,o,x],[x,o,o]]
```

does (the middle column is a win for `o`). [8 marks]

Revise the solution to (a) to extract:

```
linewin([x,x,x]) ->

    true;

linewin([o,o,o]) ->

    true;

linewin(_) ->

     false.

 wincol(N,[First,Second,Third]) ->

     linewin([pick(N,First),pick(N,Second),pick(N,Third)]).

wincol(Board) ->

     wincol(0,Board) orelse wincol(1,Board) orelse
wincol(2,Board).
```

8 marks: Numerous solutions. Be lenient with minor syntactic deviations and with solutions that are almost there. No penalty for defensive programming.