Programming day 1 excersicesr

Formal

2.1.1 Truth tables

A	B	C	!A or !B or C
T	T	T	T
T	F	T	T
F	T	T	T
F	F	T	T
T	T	F	F
T	F	F	T
F	T	F	T
F	F	F	T

A	B	C	D	!(A and !(B or !C) or D)
T	T	T	T	F
T	F	T	T	F
F	T	T	T	F
F	F	T	T	F
T	T	F	T	F
T	F	F	T	F
F	T	F	T	F
F	F	F	T	F
T	T	T	F	T
T	F	T	F	F
F	T	T	F	T
F	F	T	F	T
T	T	F	F	T
T	F	F	F	T
F	T	F	F	T
F	F	F	F	T

2.1.2 Morgan's Law

- !(A and !(B or !C) or D)
    - !(A and !B and C or D)
        - Solution: !A or B or !C and !D
- !(!(!A and B) and !(C or !D))
    - !(A or !B and !C and D)
        - Solution: !A and B or C or D
- A or (C and !(B or C))
    - A or C and (!B and !C)
        - Solution: A or C and !B and !C (will always reflect the value of A)

2.1.3 Simplify expression

- A and (!A and B)
    - Solution: false
- A and (!A or B)
    - Solution: A and B
- (A and !B) or (!A and B)
    - Solution A xor B
- (A or !B) and (!A or B)
    - Solution: A and B
- !(A and !B) or (A and !B)
    - !A or B or A and !B Solution: A and !B

2.2.1.1 Control structures conditions

```
if(customer.hasKid){
    if(customer.kid.age < 5){
        show(bunnies);
    }else if(customer.kid.age <= 10){
        show(mice)
    }else{
        // we could move the condition into the else but to clarify that we test a
different property check conditions within the sle block
        if(customer.kid.gender == f){
            show(rats)
        }else{
            show(kitten);
        }
    }
}else if(custom.IsAlone){
    show(puppies);
}else{
    show(fish)
}
```

2.2.1.2 Control structures case matching

```
       Check type of animals
        /                   \
    fish                   rodent
   /     \               /     | \
tropical  \            rat     |  mice
        goldfish                |
                              hamster
```

Decision tree has 2 layers, which makes just one switch statement unsuiteable. Pseudo code reflects these 2 layers

```
switch(animal family)
    case(rodents){
        switch(species){
            case(rat): feed(nuts, meat)
            case(hamster): feed(grain)
            case(mice): feed(nuts)
        }
    }
    case(fish){
       if(fish.isTropical  == true){
          feed(dry food);
       } else if(fish.isGoldfish){
          feed(frozen food);
       }
    }
```

2.2.1.3 Control structures loops

Diagram *(maybe not the best decision to NOT use a drawing tool in hindsight)*

```
Start loop

--------------
| get item   |
--------------
   |
   /\
  /  \
  \  / is parcel
   \/\
   |  \ yes
no |   \
   |   -------------
   |   | send left |
   |   -------------
   |             |
```

```
--------------    |
| send right |    |
--------------    /
  |              /
  |     _ _ _ _ /
  /\  /
 /  \/
 \  / items left (if yes, go back to start)
  \/

 End loop
```

```
loop{
    var item = getNextItem(); // in "real" code, i would probably do something
like "while(itemList.next != null))...."
    if(item.isParcel){
        send(item, right);
    }
    else{
        send(item, left);
    }
    if(!hasMoreItems()){
        exit loop
    }
}
```

2.2.2.1 Constants and variables

Not sure i moved, I understood how to shorten code execution. Did the following things:

- moved the first multiplication in the variable assignments (would break edge case when input is 1)
- skipped one loop round by already exiting the loop when counter is 2 (as multiplying with 1 makes no sense) -> this might by what reduces the nof steps according to task)
- reversed the counter order (decrementing instead of incrementing). Should have no influence

```
var result= input * input-1;
var counter = input-2;
loop
    result = result * counter;
    if(counter =< 2){
        exit loop;
    }
    counter = counter -1;
end loop
```

2.2.2.2 arrays

```
const input = [2, 3,4]; // random array
var result = 0;
var counter = 0;
loop
    result = result + input[counter];
    counter++;
end loop
```

2.2.2.3 map

```
const input = [];
const corners = {
    "Triangle": 3,
    "Rectangle": 4,
    "Pentagon": 5
    "Hexagon": 6
}
var counter = 0;
Loop
    var key,value = corners[counter];
    input[counter] = value; // assuming this works (some languages would throw an
ex as array are defined as fixex)
    counter++;
End loop
```

Usually, it is possible to iterate over the entries in a map/dic, but quite possible the idea was to access the map values explicitly via key. Example: `corners["Pentagon"]`

- not sure if I understand the exercise at all 😦
    - add the numbers INTO an array? (creating the array?...assuming it is not a fixed size array)
    - create the sum of the nof corners passed as an array (where is the map?)?
    - where

2.2.3.1 methods

```
// to be called like: sum([1,2,3,4,5])
method sum = (input : []) => { // method as we want a return value
    var result = 0;
    var counter = 0;
    loop
        result = result + input[counter];
        counter++;
    end loop¨
    return result;
}
// combine
```

```
const array = [3, 4];
const calculatedSum = sum(array); // 7
const factorialSum = calculateFactorial(calculatedSum);

// user input
const userInput : number // assuming dataType is number (otherwise we would have
to convert the user input)
const factorialSum = calculateFactorial(userInput);
```

Regular expressions

Remark: tested using PCRE2 from https://regex101.com/ (Could be that sometimes the "/" as prefix and suffix is missing as this is not required by regex101)

3.1.1 Literal matching Text: "This is nice little text" Search for "nice" using regex: `/nice/`

Problem: this would match false positives such as nicer or it would not match "Nice". We cannot use " " (`/ nice /`) cause we would not find all "nice" (for example when in brackets, end of sentence, line break etc.)

3.1.2 Alternatives Regex: `/if|then|else/` I didn't find any false positives (unwanted matches) with code from the formal languages chapter. However, regex would match words like "lift", "elsewhere" etc.

3.1.3 Word boundaries Regex: `/\bif\b|\bthen\b|\belse\b/`

3.1.4 White space `/\sif\s/` would not find an if at the beginning of a text

3.1.5 Character classes Regex for nit, wit, kit, lit: `/[nwkl]it/` (or `/\b[nwkl]it\b/` with word boundaries) Regex for Aloof, aloof, AlooF, alooF: `/[Aa]loo[fF]/`

3.1.6 Quantifiers Task: "?", "+" "*" using between m and n notation

- "?": `{0,1}`
- "+": `{1,}`
- "*": `{0,}`

3.1.7 Wildcard `/\b[0-9]+\s.+\b/` -> matches: "23 pigs" or "3323 turtles"

3.1.8 Negation `[^0-9]*` -> anything except lower case characters `[^a-z]*` -> anything except lower case characters

Remark: in the concept the caret is outside of the brackets. Shouldnt it be inside?

3.1.9 Escaping characters `true \/ false price \[EUR\]`

3.1.10 Groups Windows path: Starts with drive letter, followed by ":", each folder has a delimiter "/" and folder name a limited set of characters. Regex: `/[cCdDcE]:(?:[\\][0-9a-zA-Z-_]+){0,}[\\]?/` *(Some aspects like correct drive letters, "/" as delimiter or excact set of characters were ignored for the sake of simplicity)*

Knock knock joke (starts with "knock knock",line 2: "who's there, line 3: name, line 4 contains "{name}" again.) Regex: `Knock, knock!\nWho's there\?\n(.+).\n.*\1.*\n.*\1.*` *(Some aspects like upper/lower case or*

punctuation were just assumed for simplicity) Regex with named groups: `Knock, knock!\nWho's there\?\n(?'name'.+).\n.*\g{name}.*\n.*\g{name}.*`

3.1.11 Anchors and multiline

- `/^ | $/gm` matches any whitspace at the beginning (deliberatyl not using "\s" as we only want whitespace)
- `^ +$/gm` matches lines only consisting of spaces

Could it be that the example in the concept (chapter 3.4.11) "$myself^" is incorrect and never matches anything. Shouldnt it be "^myself$"

3.1.2 Lookahead/Lookbehind `(?=\s)|(?<=\s)` wordboundary (like "\b") -> example: `/(?=\s)|(?<=\s)hello(?=\s)|(?<=\s)/gm` -> will find all "hello" in a text but only if hello starts/ends with a whitespace (or linebreak etc.)

3.1.3 Delimiters `/(?:[\/][0-9a-zA-Z-_]+){0,}[\/]?/gm` Unix path with normal delimiter (similar to windows paths) `%(?:[/][0-9a-zA-Z-_]+){0,}[/]?%` -> same thing but using % as delimiter

3.2.1 Replacing `/(\bif.+then(?!=\b))|([^\n]+(?=else))|(else(?!=\b))|([^\n]+(?=end if))/gm` Replace: `$1$2$3$4\n`-> will format code with then/else and end if (there are probably more elegant solutions)

- same thing in one: `/(\bif.+then(?!=\b))|([^\n]+(?=else))|(else(?!=\b))|([^\n]+(?=end if))/$1$2$3$4\n/gm`

Https url (*url regex is simplified for readability* `/(((?<=http:\/\/)|(?<=https:\/\/))[a-zA-Z0-9][a-zA-Z0-9-]+[a-zA-Z0-9]\.[^\s]{2,})/gm` -> substitution `<a href="https://$1">link to $1</a>`

3.2.2 Case Conversion Regex converting condition keywords uppercase:

```
/(\bif\b|\bthen\b|\belse\b|\bend\b)/\U$1/gmi
```