1. Describe the expression \((a - b \times c) \times (c \times d - e) + f - (g - h)\) in Lisp's prefix form.

\[
(\text{+} (\text{*} (\text{-} a (\text{*} b c))) (\text{-} (\text{*} c d) e)) (\text{-} f (\text{-} g h))
\]
2. Use bottom-up algorithm to parse the expression from Problem 1, build the dependency graph, explain which operations can be performed in parallel first, which next, etc.

\[(a - b \times c) \times (c \times d - e) + f - (g - h)\]

\[
\begin{align*}
\text{step 1:} & \quad r_1 \leftarrow b \times c \\
\text{step 2:} & \quad r_2 \leftarrow a - r_1 \\
\text{step 3:} & \quad r_3 \leftarrow c \times d \\
\text{step 4:} & \quad r_4 \leftarrow r_3 - e \\
\text{step 5:} & \quad r_5 \leftarrow g - h \\
\text{step 6:} & \quad r_6 \leftarrow f - r_5 \\
\text{step 7:} & \quad r_7 \leftarrow r_2 \times r_4 \\
\text{step 8:} & \quad r_8 \leftarrow r_7 + r_6
\end{align*}
\]
3. Describe what will be the quadruples generated by Java based on the following program segment:

```java
for(int i = 1; i <= n; i++)
    {c[i] = a[i] + b[i];}

L3: i = i + 1
if (r < i <= n) 
    if (r) else go to L2;
L1: c[i] = a[i] + b[i]
i = i + 1
    go to L3
L2:
```

Yes, we can use parallel-for in computations of #3 because the computations are parallelizable.
5. Use the algorithm from Problem 3 to write a generic Java method for computing the component-wise sum of two arrays. This method should work for all possible numerical data types: int, short, char, double, float.

```java
public static int[] compWiseSum(int[] a, int[] b) {
    int[] c = new int[a.length];
    for (int i = 0; i < c.length; i++) {
        c[i] = a[i] + b[i];
    }
    return c;
}
```
6. Write a LISP function for computing the component-wise sum of two lists. Trace it on the example of the lists ’(1 10 100) and ’(2 20 200), the result should be ’(3 30 300).

```
(defun cws (list1 list2)  
  (if (null list1) list2)  
  (+ (car list1) (car list2))  
  (cws (cdr list1) (cdr list2)))
```

```
(cws ’(1 10 100) ’(2 20 200))  
(cons (+ 1 2) (cws ’(10 100) ’(20 200)))  
(cons (+ 1 2) (cons (+ 10 20) (cws ’(100) ’(200))))  
(cons (+ 12) (cons (+ 10 20) (cws ’(100 200) ’(120))))  
(cons (+ 12) (cons (+ 10 20) (cws ’(100 200) ’(130))))  
(cons (+ 12) (cons (+ 10 20) ’(300)))  
’(3 30 300)
```
7. Write a Prolog program for computing the component-wise sum of the two lists. Trace it on the same example as in Problem 5.

```
cws([3, 3, 3, 3])
cws([[1, T], [H2, T2]], [H, T]) ← sumOfLists([T1, H2], T)
   H is H1 + H2

   cws([1 10 100, 2 20 200], What)
   H1 = 1  T1 = [10, 100]
   H2 = 3  T2 = [20, 200]
   H is H1 + H2 = 3

   cws([10 100], [20 200], T)
   H1' = 10  T1' = [100]
   H2' = 20  T2' = [200]
   H is H1' + H2' = 30

   cws([100, 3], [2003], T')
   H1'' = 100
   H2'' = 200
   H'' = 200
   T' = empty
```
8. We all know how to find the largest element of an array. Use Algol's call-by-name feature to write a method `max(index, lower, upper, expression)` that would be able to compute the largest value of a given expression when the index of this expression ranges between the given lower bound and the given upper bound. Show, step by step, how this method can be used to find the largest value of the expression \((0.1 \cdot j) - \sqrt{0.1 \cdot j}\), when \(j\) ranges from 0 to 10.

```java
public static max(int index, int lower, int upper, int expr)
{
    aux = MIN_VALUE;
    for (index = lower; index <= upper; index++)
    {
        largest = expr;
        if (largest > aux)
        {
            aux = largest;
        }
    }
    return aux;
}

Max(j, 0, 10, (0.1 * j) - \sqrt{0.1 * j})

for (j = 0; j <= 10; j++)
{
    largest = (0.1 * j) - \sqrt{0.1 * j}
    if (largest > aux)
    {
        aux = largest;
    }
    return aux;
}
```
9. Use the predicates parent, male, and female to describe the concept of a brother. Test your definition on the following database related to the Russian czar Peter I, the founder of St. Petersburg, Russia:

```
parent(alexei, peter).
paren(alexei, ivan).
parent(alexei, sofia).
male(peter).
male(ivan).
female(sofia).
? brother(X, Y).
```

The goal is to find all the brothers. Show, step by step, how Prolog will do it.

```
brother(X, Y) :- male(X), parent(P, X), parent(P, Y), X != Y.

?- brother( X, Y).
```

```
brother(Peter, Ivan)

male(Peter)
paren(Peter, Alexei)
paren(Alexei, Ivan)
```

```
brother(Peter, Sofía)

male(Ivan)
paren(Ivan, Alexei)
paren(Alexei, Sofía)
```

```
(Peter, Ivan)
(Peter, Sofía)
```

file:///Q:/cs3360.17/test2.html
10. Use wave algorithm to solve the following problem:

- we know that distance $d$ is equal to velocity $v$ multiplied by time $t$: $d = v \times t$;
- we know that the cost $c$ of the gas is proportional to the amount of fuel: $c = 2.10 \times f$;
- we know that the amount of fuel $f$ is proportional to the distance: $f = d / 20$.

We know the velocity $v$ and the time $t$. We need to compute the cost $c$ of the gas.

\[
\begin{align*}
K_d & \leftarrow K_u, K_t \\
K_u & \leftarrow K_d, K_t \\
K_t & \leftarrow K_u, K_d \\
K_c & \leftarrow K_f \\
K_f & \leftarrow K_c \\
K_f & \leftarrow K_d \\
K_d & \leftarrow K_f
\end{align*}
\]