1. Write a function `median` that computes the median of a given list of numbers. For example, if \([1,5,0]\) is given as input, the function should return 1. For input \([3,5,0,2]\), the output should be 2.5.

2. Write a function `insertPos` that takes an element \(x\) and a list \(y\) as input parameters. It will insert the element \(x\) into all possible positions of \(y\). The result should be a list containing all the lists with \(x\) inserted into different positions. For example, the input 2 and \([3,5]\) should return \([[2,3,5],[3,2,5],[3,5,2]]\).

3. Write a function `innerprod` that takes two vectors \(v\) and \(w\) represented by lists and returns the inner product. The inner product of two vectors is defined as \(\vec{v} \cdot \vec{w} = v_1 \cdot w_1 + v_2 \cdot w_2 + \ldots v_n \cdot w_n\). For example, if \(v = [3,5,0,2]\) and \(w = [2,3,1,4]\), then the product of \(v\) and \(w\) is equal to \(3 \cdot 2 + 5 \cdot 3 + 0 \cdot 1 + 2 \cdot 4 = 6 + 15 + 0 + 8 = 29\). The input vectors must have equal length. Otherwise the return value must be -1.

4. Implement the sieve of Eratosthenes in Haskell. This algorithm determines all the prime numbers in a range of numbers by removing all the multiples of 2,3,5,... from the range. What is left in the range are only prime numbers. For example, for the range \([2..20]\) (1 is not a prime number), we would first remove all multiples of 2 and are left with \([2,3,5,7,9,11,13,15,17,19]\). In the next step we remove multiples of 3 and are left with \([2,3,5,7,11,13,17,19]\). Once we reach a number whose first multiple is larger than 20, we stop.

5. (a) Write a function `qs_sorted` which employs the quicksort algorithm (see exercise sheet 6), gets a list of lists as an input and returns a list of lists as an output, such that each outputted list is sorted using the values of the last list as sort keys. For example, the input

\([[0,1,2],[23,26,30],[3400,1700,5000]]\)

should result in the output:

\([[1,0,2],[26,23,30],[1700,3400,5000]]\).
(b) Write two functions `qs_lc_tuple_f` and `qs_lc_tuple_l` which employ the quicksort algorithm, get a list of tuples of the form `(Int,Int)` and return the list sorted in ascending order of the first element or the last element respectively.

For example,

```
qs_lc_tuple_f [(5,1),(6,4),(2,8),(4,2)]
[(2,8),(4,2),(5,1),(6,4)]
```

and

```
qs_lc_tuple_l [(5,1),(6,4),(2,8),(4,2)]
[(5,1),(4,2),(6,4),(2,8)]
```

You can use the build-in functions: `minimum`, which returns the minimum element of a list, and `delete`, which deletes the first occurrence of an element from a list. In order to use `delete` you have to import the module `Data.List`.