

# fn function\_report\_top\_k

Michael Shoemate

March 18, 2025

## 1 Hoare Triple

### Precondition

#### Compiler-verified

Types consistent with pseudocode.

#### Caller-verified

None

### Pseudocode

```
1 def function_report_top_k(k: int, optimize: str) -> Callable[[list[TIA]], list[int]]:
2     def function(x: list[TIA]) -> int:
3         if optimize == "max": #
4             cmp = lambda a, b: a > b
5         else:
6             cmp = lambda a, b: a < b
7
8         def max_sample(a, b): #
9             return a if cmp(a[1], b[1]) else b
10
11        return [i for i, _ in top(x, k, max_sample)] #
12
13    return function
```

### Postcondition

**Theorem 1.1.** Returns a noninteractive function with no side-effects that, when given a vector of scores  $x_i$ , returns the indices of the top  $k$   $y_i$ , where each  $y_i = -x_i$  if **optimize** is **min**, else  $y_i = x_i$ .

The returned function will only error if there are no non-null scores.

*Proof.* The comparator on line 3 effectively flips the sign of the scores if **optimize** is **min**, therefore each  $y_i = -x_i$  if **optimize** is **min**, else  $y_i = x_i$ . This is done to avoid the negation of a signed integer.

Assume the scores are non-null, as required by the returned function precondition. Then **max\_sample** on line 8 defines a total ordering on the scores.

Since the score vector is finite, and **max\_sample** defines a total ordering, then the preconditions for **top** are met. Therefore on line 11 **top** returns the pairs with the top  $k$  scores. Line 11 then discards the scores, returning only the indices, which is the desired output.

There is one source of error in the function, when there are no non-null scores in the input vector.  $\square$