# The Flix Cheetsheet

## Opt

An option Opt is either None or Some(v). *Options cannot be nested[[1]](#footnote-1).*

|  |
| --- |
| null(o: Opt[A]): Bool  Returns true iff o is None. |
| get(o: Opt[A]): A [Unsafe]  Returns v if o is Some(v). Otherwise the semantics is undefined. |
| getWithDefault(o: Opt[A], a: A): A  Returns v if o is Some(v). Otherwise returns a. |
| exists(f: A => Bool, o: Opt[A]): Bool  Returns the value of f(v) if o is Some(v). Otherwise false. |
| forall(f: A => Bool, o: Opt[A]): Bool  Returns the value of f(v) if o is Some(v). Otherwise true. |
| filter(f: A => Bool, o: Opt[A]): Opt[A]  Returns o if o is Some(v) and f(v) is true. Otherwise returns None. |
| map(f: A => B, o: Opt[A]): Opt[B]  Returns Some(f(v)) if o is Some(v). Otherwise returns None. |
| map2(f: (A, B) => C, o1: Opt[A], o2: Opt[B]): Opt[C]  Returns Some(f(v1, v2)) if o1 is Some(v1) and Some(v2).  Otherwise returns None. |
| flatMap(f: A => Opt[B], o: Opt[A]): Opt[B]  Returns f(v) if o is Some(v). Otherwise returns None. |
| flatMap2(f: (A, B) => Opt[C],  o1: Opt[A], o2: Opt[A]): Opt[C]  Returns f(v1, v2) if o1 is Some(v1) and o2 is Some(v2).  Otherwise returns None. |
| toList(o: Opt[A]): List[A]  Returns a one-element list of the value v if o is Some(v).  Otherwise returns the empty list. |
| toSet(o: Opt[A]): Set[A]  Returns a one-element set of the value v if o is Some(v).  Otherwise returns the empty set. |
| withDefault(o1: Opt[A], o2: Opt[A]): Opt[A]  Returns o1 if it is Some(v) otherwise returns o2. |

## List

A list is either the empty list Nil or a cons cell v :: vs.

### Basic Operations

|  |
| --- |
| Nil: List[A]  The empty list. |
| v :: vs: List[A]  The list with the element v followed by the list vs. |
| null(xs: List[A]): Bool  Returns true iff xs is the empty list, i.e. Nil. |
| head(xs: List[A]): A [Unsafe]  Returns the first element of the list.  If the list is empty the semantics are undefined. |
|  |
| range(Int, Int)  Returns a list |
| repeat(A, Int) |
| permutations(List[A]): List  fdsfds |
| null(xs: List[A])  Returns true if xs is the empty list. |
| map(f: A => B, xs: List[A]): List[B]  Foo |
| flatMap(f: A => List[B], xs: List[A]): List[B] |
|  |
|  |

## Set

## Map

1. This ensures efficient representation of options. [↑](#footnote-ref-1)