AI assignment one.

Assumptions made:

1. The field may have various size, this is regulated through the boundaries/2 predicate in the main file.
2. It is allowed to make blind passes.
3. A pass is aimed at one specific spot on a line or diagonal, not the whole line, so a ball can be passed to any human on the line, not just the first one.
4. A pass is successful if there is a human at the point, where a pass is aimed at and there are no orcs on the line or diagonal between the origin of pass and the target point.
5. Backtracking is allowed after failing any move (e.g getting tackled by an orc or failing a pass).
6. 1 attempt is counted as a series of moves of arbitrary length which ends either at touchdown or at orc.

Some key points:
All the code can be viewed at https://swish.swi-prolog.org/p/hmUMAPpc.swinb or in the source file attached below. Note that the code is slightly different on the site and in source in particular:

1. Input method for web version is the first cell of the notebook, where the predicates are specified. Source version reads the file “input.pl” for input.
2. If you choose to run the source code I suggest you run “swipl assignment.pl” and then paste the queries from web version since some maps may cause nontermination and it is impractical to run all three algorithms at once.
3. My version of search algorithm uses predicate findall/3, so it is not runnable in the web version :(
4. Statistical analysis for algotithms is also provided at https://swish.swi-prolog.org/p/hmUMAPpc.swinb
5. A hand off play does not count towards the overall number of steps, however, it is mentioned in the output with its own tag to allow more comfortable checking.

Description of my algorithm.

The algorithm consists of three phases. First, it traverses all the points that are reachable without passing and remembers the lowest number of steps required to reach that node, similar to Dijkstra's algorithm. Than it tries to expand the domain of points traversed by making passes from all the known points to see if new points become accessible. Phase three is phase one, but the set of initial points is set of points found in phase two. For each point the algorithm remembers its previous point to be able to construct the path. Algorithm stops as soon as it finds touchdown point to increase average time performance, although it leads to worse path lengths. The algorithm guarantees to find a path in relatively little time as

long as the path exists. However, it does not pass unless it is necessary to solve a map so it does not perform to well on easy maps.

Maps tested.

impossible.pl:

```
5. . . . . .
4. . . . . .
3. . . . . .
2. o o o . .
1. o h o . .
0. o t o . .
 0 1 2 3 4 5
```

There is no reachable touchdown on this map so algorithms should output false, however backtracking algorithm will result with nontermination since it tries all possible paths, whos number grow really fast with the size of the field.

hard-pass.pl:

```
5. . . . . .
4. . . . . .
3. . . . . .
2. o o . . .
1. o h o . .
0. o t o . .
 0 1 2 3 4 5
```

This map is only solvable with a pass to a specific location, so it requires algorithms to keep their only pass until it is necessary.

maze.pl:

```
5o o o o o o
4. . . o . t
3. o . o . .
2. o . o . .
1. o . o . .
0. o . . . .
 0 1 2 3 4 5
```

requires precise movement, so it is hard for random search method to pass.

Random maps!

Algorithms have been tested on several random maps with statistical analysis based on the results of passing those.

Does vision matter?

No. None of the algorithms relies on using the player’s vision range since they consider all possible moves and blind passes.