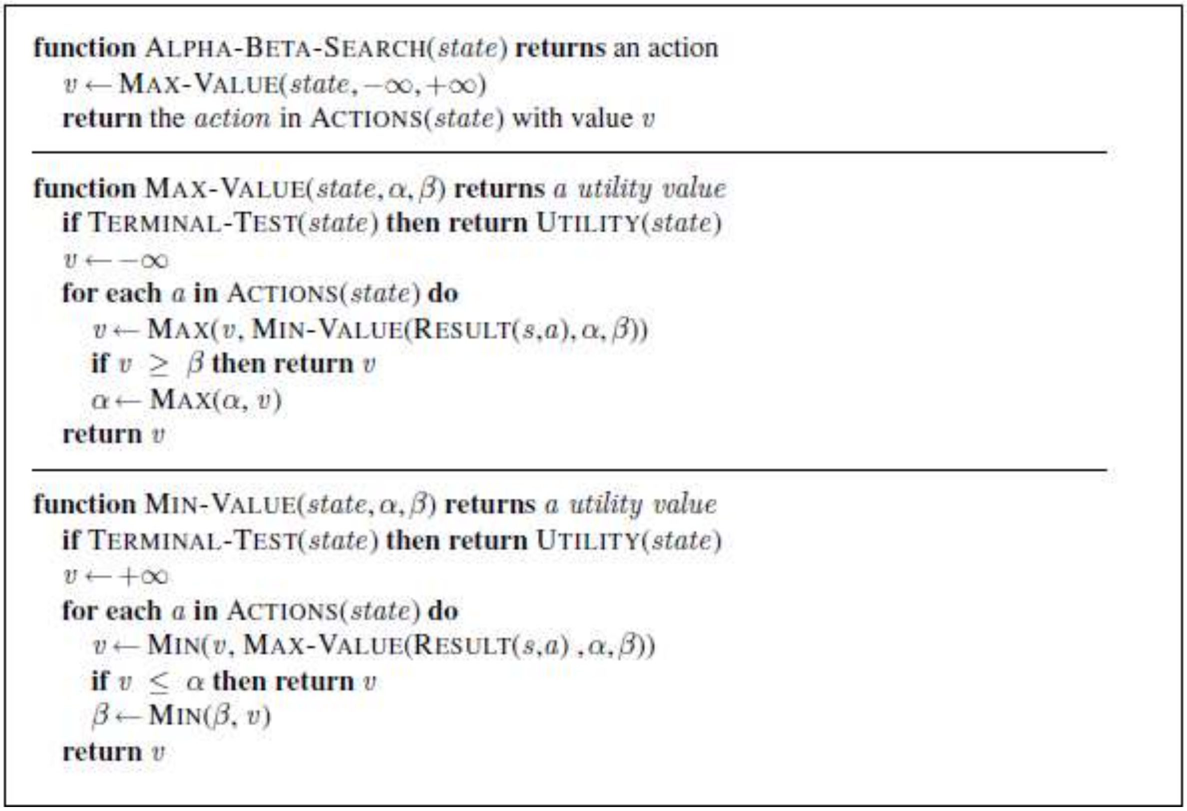
LAB 9 Adversarial Search – II

1.1 Alpha-Beat Pruning:

In some cases, it is extremely useful to be able to **prune** sections of the game tree. Using alpha–beta pruning, it is possible to remove sections of the game tree that are not worth examining, to make searching for a good move more efficient. The principle behind alpha–beta pruning is that if a move is determined to be worse than another move that has already been examined, then further examining the possible consequences of that worse move is pointless.



```
function ALPHA-BETA-SEARCH($state$) returns an action
   $v \leftarrow$ MAX-VALUE($state, -\infty, +\infty$)
   return the $action$ in ACTIONS($state$) with value $v$

function MAX-VALUE($state, \alpha, \beta$) returns $a$ $utility$ $value$
   if TERMINAL-TEST($state$) then return UTILITY($state$)
   $v \leftarrow -\infty$
   for each $a$ in ACTIONS($state$) do
      $v \leftarrow$ MAX($v$, MIN-VALUE(RESULT($s$,$a$), $\alpha, \beta$))
      if $v \geq \beta$ then return $v$
      $\alpha \leftarrow$ MAX($\alpha$, $v$)
   return $v$

function MIN-VALUE($state, \alpha, \beta$) returns $a$ $utility$ $value$
   if TERMINAL-TEST($state$) then return UTILITY($state$)
   $v \leftarrow +\infty$
   for each $a$ in ACTIONS($state$) do
      $v \leftarrow$ MIN($v$, MAX-VALUE(RESULT($s$,$a$) , $\alpha, \beta$))
      if $v \leq \alpha$ then return $v$
      $\beta \leftarrow$ MIN($\beta$, $v$)
   return $v$
```

Figure 0-1. The alpha–beta search algorithm. Notice that these routines are the same as the MINIMAX functions except for the two lines in each of MIN-VALUE and MAX-VALUE that maintain α and β (and the bookkeeping to pass these parameters along).

1.2 Lab Tasks

Exercise 9.1.

Modify the program to implement Tic-Tac-Toe game from LAB 08 to incorporate alpha-beta pruning.