```
PriorityRegAlloc(proc, regCount) {
   ig ← buildInterferenceGraph(proc)
   unconstrained ←
      { n ∈ nodes(ig) | neighborCount(n) < regCount }
   constrained ←
      { n ∈ nodes(ig) | neighborCount(n) ≥ regCount }

   while( constrained ≠ ϕ )  {
      for ( c ∈ constrained such that not colorable(c)
            and canSplit(c) ) {
         c1, c2 ← split(c)
         constrained ← constrained - {c}
         if ( neighborCount(c1) < regCount )
               unconstrained ← unconstrained ∪ { c1}
         else  constrained ← constrained ∪ {c1}
         if ( neighborCount(c2) < regCount )
               unconstrained ← unconstrained ∪ { c2}
         else  constrained ← constrained ∪ {c2}
         for ( d ∈ neighbors(c) such that
               d ∈ unconstrained and
                neighborCount(d) ≥ regCount ){
                 unconstrained ← unconstrained - {d}
                constrained ← constrained ∪ {d}
      }     } // End of both for loops
```

```
    /* At this point all nodes in constrained are
        colorable or can't be split */

    Select p ∈ constrained such that
                priority(p) is maximized
     if ( colorable(p) )
            color(p)
     else  spill(p)
  } // End of While
 color all nodes ∈ unconstrained
}
```