Preemptable\_Queue\_Template - Dylan

|  |  |  |
| --- | --- | --- |
| Operation Enqueue(alters E: Entry; updates Q: P\_Queue); | | |
| Valid Inputs | Expected Outputs | Reason |
| #E = 5  #Q = <1, 2, 3, 4> | E = <>  Q = <1, 2, 3, 4, 5> | Boundary,  assuming Max\_Length = 5 |
| #E = 1  #Q = <> | E = <>  Q = <1> | Boundary |
| #E = 3  #Q = <1, 2> | E = <>  Q = <1, 2, 3> | Routine |

|  |  |  |
| --- | --- | --- |
| Operation Inject(alters E: Entry; updates Q: P\_Queue); | | |
| Valid Inputs | Expected Outputs | Reason |
| #E = 0  #Q = <1, 2, 3, 4> | E = <>  Q = <0, 1, 2, 3, 4> | Boundary,  assuming Max\_Length = 5 |
| #E = 1  #Q = <> | E = <>  Q = <1> | Boundary |
| #E = 0  #Q = <1, 2> | E = <>  Q = <0, 1, 2> | Routine |

|  |  |  |
| --- | --- | --- |
| Operation Dequeue(replaces R: Entry; updates Q: P\_Queue); | | |
| Valid Inputs | Expected Outputs | Reason |
| #R = <>  #Q = <0, 1, 2, 3, 4> | R = 0  Q = <1, 2, 3, 4> | Routine |
| #R = <>  #Q = <0> | R = 0  Q = <> | Boundary |
| #R = <>  #Q = <0, 1, 2> | R = 0  Q = <1, 2> | Routine |

|  |  |  |
| --- | --- | --- |
| Operation Swap\_Last\_Entry(updates E: Entry; updates Q: P\_Queue); | | |
| Valid Inputs | Expected Outputs | Reason |
| #E = <1>  #Q = <1, 2, 3, 4> | E = <1, 2, 3, 0>  Q = <4> | Routine |
| #E = <0>  #Q = <1> | E = <1>  Q = <0> | Boundary |
| #E = <9>  #Q = <1, 2> | E = <2>  Q = <1, 9> | Routine |

|  |  |  |
| --- | --- | --- |
| Operation Length(restores Q: P\_Queue): Integer; | | |
| Valid Inputs | Expected Outputs | Reason |
| #Q = <> | Q = <>  Length() = 0 | Boundary |
| #Q = <1> | Q = <1>  Length() = 1 | Routine |
| #Q = <0, 1, 2, 3, 4, 5, 6, 7> | Q = <0, 1, 2, 3, 4, 5, 6, 7>  Length() = 8 | Routine |

|  |  |  |
| --- | --- | --- |
| Operation Rem\_Capacity(restores Q: P\_Queue): Integer; | | |
| Valid Inputs | Expected Outputs  (Max\_Length =5) | Reason |
| #Q = <> | Q = <>  Rem\_Capacity() = 5 | Boundary |
| #Q = <1, 2, 3, 4, 5> | Q = <1, 2, 3, 4, 5>  Rem\_Capacity() = 0 | Boundary,  Assuming Max\_Length = 5 |
| #Q = <1, 2, 3, 4> | Q = <1, 2, 3, 4>  Rem\_Capacity() = 1 | Routine |

|  |  |  |
| --- | --- | --- |
| Operation Clear(clears Q: P\_Queue); | | |
| Valid Inputs | Expected Outputs | Reason |
| #Q = <> | Q = <> | Boundary |
| #Q = <1> | Q = <> | Routine |
| #Q = <1, 2, 3, 4, 5> | Q = <> | Boundary, assuming Max\_Length = 5 |

Globally\_Bounded\_List\_Template - Jesse

|  |  |  |
| --- | --- | --- |
| Oper Advance( upd P: List ); | | |
| Valid Inputs | Expected Outputs | Reason |
| #P.Rem = <0, 1, 2>  #P.Prec = <7, 8, 9> | P.Rem = <1, 2>  P.Prec = <7, 8, 9, 0> | Routine |
| #P.Rem = <0, 1, 2>  #P.Prec = <> | P.Rem = <1, 2>  P\_Prec = <0> | Boundary |
| #P.Rem = <0>  #P.Prec = <1, 2, 3> | P.Rem = <>  P\_Prec = <1, 2, 3, 0> | Boundary |

|  |  |  |
| --- | --- | --- |
| Oper Reset( upd P: List ); | | |
| Valid Inputs | Expected Outputs | Reason |
| #P.Rem = <0, 1, 2>  #P.Prec = <> | P.Rem = <0, 1, 2>  P.Prec = <> | Boundary |
| #P.Rem = <3, 4, 5>  #P.Prec = <0, 1, 2> | P.Rem = <0, 1, 2, 3, 4, 5>  P\_Prec = <> | Routine |
| #P.Rem = <>  #P.Prec = <1, 2, 3> | P.Rem = <1, 2, 3>  P\_Prec = <> | Boundary |

|  |  |  |
| --- | --- | --- |
| Oper Is\_Rem\_Empty( rest P: List ): Boolean; | | |
| Valid Inputs | Expected Outputs | Reason |
| #P.Rem = <2>  #P.Prec = <> | P.Rem = <2>  P.Prec = <>  Is\_Rem\_Empty() = False | Boundary |
| #P.Rem = <>  #P.Prec = <> | P.Rem = <>  P.Prec = <>  Is\_Rem\_Empty() = True | Boundary |
| #P.Rem = <>  #P.Prec = <2> | P.Rem = <>  P.Prec = <2>  Is\_Rem\_Empty() = True | Boundary |

|  |  |  |
| --- | --- | --- |
| Oper Insert( alt New\_Entry: Entry; upd P: List ); | | |
| Valid Inputs | Expected Outputs | Reason |
| #P.Rem = <1, 2, 3>  #P.Prec = <>  #New\_Entry = <0> | P.Rem = <0, 1, 2, 3>  P.Prec = <>  New\_Entry = <> | Routine |
| #P.Rem = <>  #P.Prec = <9>  #New\_Entry = <0> | P.Rem = <0>  P.Prec = <9>  New\_Entry = <> | Boundary |
| #P.Rem = <9>  #P.Prec = <>  #New\_Entry = <8> | P.Rem = <8, 9>  P.Prec = <>  New\_Entry = <> | Routine |

|  |  |  |
| --- | --- | --- |
| Oper Remove( rpl Entry\_Removed: Entry; upd P: List ); | | |
| Valid Inputs | Expected Outputs | Reason |
| #P.Rem = <1, 2, 3>  #P.Prec = <>  #Entry\_Removed = <> | #P.Rem = <2, 3>  #P.Prec = <>  #Entry\_Removed = <1> | Routine |
| #P.Rem = <1>  #P.Prec = <5>  #Entry\_Removed = <> | #P.Rem = <>  #P.Prec = <5>  #Entry\_Removed = <1> | Boundary |
| #P.Rem = <0, 1>  #P.Prec = <5>  #Entry\_Removed = <> | #P.Rem = <1>  #P.Prec = <>  #Entry\_Removed = <0> | Routine |

|  |  |  |
| --- | --- | --- |
| Oper Advance\_to\_End( upd P: List ); | | |
| Valid Inputs | Expected Outputs | Reason |
| #P.Rem = <>  #P.Prec = <0, 1, 2> | P.Rem = <>  P.Prec = <0, 1, 2> | Boundary |
| #P.Rem = <3, 4, 5>  #P.Prec = <0, 1, 2> | P.Rem = <>  P\_Prec = <0, 1, 2, 3, 4, 5> | Routine |
| #P.Rem = <1, 2, 3>  #P.Prec = <> | P.Rem = <>  P\_Prec = <1, 2, 3> | Boundary |

|  |  |  |
| --- | --- | --- |
| Oper Swap\_Remainders( upd P, Q: List ); | | |
| Valid Inputs | Expected Outputs | Reason |
| #P.Rem = <>  #P.Prec = <0, 1, 2>  #Q.Rem = <3, 4, 5>  #P.Prec = <3, 4, 5> | P.Rem = <3, 4, 5>  P.Prec = <0, 1, 2>  Q.Rem = <>  P.Prec = <3, 4, 5> | Boundary |
| #P.Rem = <3, 4, 5>  #P.Prec = <0, 1, 2>  #Q.Rem = <>  #P.Prec = <3, 4, 5> | P.Rem = <>  P.Prec = <0, 1, 2>  Q.Rem = <3, 4, 5>  P.Prec = <3, 4, 5> | Boundary |
| #P.Rem = <6, 7, 2>  #P.Prec = <9, 4, 1>  #Q.Rem = <7, 7, 1>  #P.Prec = <9, 0, 1> | P.Rem = <7, 7, 1>  P.Prec = <9, 4, 1>  Q.Rem = <6, 7, 2>  P.Prec = <9, 0, 1> | Routine |

|  |  |  |
| --- | --- | --- |
| Oper Is\_Prec\_Empty( rest P: List ): Boolean; | | |
| Valid Inputs | Expected Outputs | Reason |
| #P.Rem = <>  #P.Prec = <2> | P.Rem = <>  P.Prec = <2>  Is\_Rem\_Empty() = False | Boundary |
| #P.Rem = <>  #P.Prec = <> | P.Rem = <>  P.Prec = <>  Is\_Rem\_Empty() = True | Boundary |
| #P.Rem = <2>  #P.Prec = <> | P.Rem = <2>  P.Prec = <>  Is\_Rem\_Empty() = True | Boundary |

|  |  |  |
| --- | --- | --- |
| Operation Clear(clears Q: P\_Queue); | | |
| Valid Inputs | Expected Outputs | Reason |
| #P.Rem = <2>  #P.Prec = <> | P.Rem = <>  P.Prec = <> | Boundary |
| #P.Rem = <>  #P.Prec = <> | P.Rem = <>  P.Prec = <> | Boundary |
| #P.Rem = <>  #P.Prec = <2> | P.Rem = <>  P.Prec = <> | Boundary |