PRIORITIZER TEMPLATE

Concept Prioritizer_Template(**type** Entry; **eval** Max_Capacity: Integer;
def const (x: Entry) $\trianglelefteq$ (y: Entry): $\mathbb{B}$);
uses Std_Integer_Fac, Std_Boolean_Fac;
requires Max_Capacity > 0 **and** Is_Total_Preordering($\trianglelefteq$);

Family Entry_Keeper $\subseteq$ **Cart_Prod**
Entry_Count: Entry$\rightarrow\mathbb{N}$;
Accepting: $\mathbb{B}$;
end;
exemplar K;
Def const Total_Entry_Ct(K: Entry_Keeper): $\mathbb{N}$ = ($\sum_{x: \text{Entry}}$ K.Entry_Count(x));
constraint Total_Entry_Ct(K) $\leq$ Max_Capacity;
initialization
ensures K.Accepting = true **and** Total_Entry_Ct(K) = 0;

Def const Is_Only_Addition(x: Entry; K1, K2: Entry_Keeper): $\mathbb{B}$ = (
K2.Entry_Count(x) = K1.Entry_Count(x) + 1 **and** $\forall$ y: Entry,
if y $\neq$ x **then** K2.Entry_Count(y) = K1.Entry_Count(y));

Oper Add_Entry(**alters** x: Entry; **updates** K: Entry_Keeper);
requires Total_Entry_Ct(K) < Max_Capacity **and** K.Accepting = true;
ensures Is_Only_Addition(@x, @K, K) **and** K.Accepting = true;

Oper Change_Modes(**updates** K: Entry_Keeper);
ensures K.Accepting = $\neg$ @K.Accepting **and** K.Entry_Count = @K.Entry_Count;

Def const (x: Entry) $\triangleleft$ (y: Entry): $\mathbb{B}$ = (x $\trianglelefteq$ y **and** $\neg$ y $\trianglelefteq$ x);

Oper Remove_a_Smallest_Entry(**replaces** s: Entry; **updates** K: Entry_Keeper);
requires K.Accepting = false **and** Total_Entry_Ct(K) > 0;
ensures K.Accepting = false **and** Is_Only_Addition(s, K, @K) **and** $\forall$ x: Entry,
if x $\triangleleft$ s **then** @K.Entry_Count(x) = 0;

Oper Total_Entry_Count(**restores** K: Entry_Keeper): Integer;
ensures Total_Entry_Count = (Total_Entry_Ct(K));

Oper Remove_Any_Entry(**replaces** x: Entry; **updates** K: Entry_Keeper);
requires Total_Entry_Ct(K) > 0;
ensures Is_Only_Addition(x, K, @K) **and** K.Accepting = @K.Accepting;

```
    Oper Is_Accepting( restores K: Entry_Keeper ): Boolean;
        ensures ⋯

    Oper Clear( clears K: Entry_Keeper );

end Prioritizer_Template;
```