Facility Queue\_Test\_Facility;

uses String\_Theory;

Facility Queue\_Test\_Fac is Queue\_Template(Char\_Str, 3)

realized by Circular\_Array\_Realiz;

Operation Write\_Part\_Queue(evaluates Num: Integer; updates Q: Queue);

requires 0 <= Num and Num <= |Q|;

Procedure

Var CS: Char\_Str;

Var I : Integer;

I := 0;

While(I < Num)

changing Q, CS, I;

maintaining |Q| = |#Q| - I and I <= |#Q|

and I >= 0;

decreasing |Q|;

do

Dequeue(CS, Q);

Write\_Line(CS);

I := I + 1;

end;

end Write\_Part\_Queue;

Operation Copy\_Queue(restores Q: Queue; replaces Q\_Copy: Queue);

requires |Q| > 0;

Procedure

Var CS1,CS2: Char\_Str;

Var I: Integer;

Var temp\_Q: Queue;

I := Length(Q);

Clear(Q\_Copy);

While(I /= 0)

changing CS1,CS2, Q, Q\_Copy,temp\_Q, I;

maintaining Q\_Copy = temp\_Q and I = |Q|

and #Q = temp\_Q o Q;

decreasing |Q|;

do

Dequeue(CS1,Q);

CS2 := CS1;

Enqueue(CS1,Q\_Copy);

Enqueue(CS2,temp\_Q);

I := Length(Q);

end;

I := Length(temp\_Q);

While(I /=0)

changing Q, temp\_Q, CS1,I;

maintaining I = |temp\_Q| and #Q = Q o temp\_Q;

decreasing |temp\_Q|;

do

Dequeue(CS1,temp\_Q);

Enqueue(CS1,Q);

I := Length(temp\_Q);

end;

end Copy\_Queue;

Operation Main();

Procedure

--Var Q: Queue;

--Enqueue("test", Q);

--Write\_Part\_Queue(1, Q);

end Main;

end Queue\_Test\_Facility;