```
/* Description:
   Here is defined the tc_EntryNoutCarSystem
   which consist in test if the system
   satisfies with the entry and out of
   car, the objetive is to feed the system
   with real datas and do a request to the system
   about one zone in one specific pCtrl and
   observer the freeSpots of this zone */


module Testbench_EntryNOutCarSystem
{
  import from declaration_Signals all;
  import from declaration_portsNComponent all;
  import from declaration_templates all;

  /* Import of Data of Entrance and Exit of Cars */

  import from In_Out_Cars all;

  /*Functions*/

  function f_EntryCar(numCtrl nCtrl,numZone nZone) runs on System
  {
      cEnv_pTesting.send(a_sEntryCarCtrl(nCtrl));
      cEnv_pTesting.send(a_sEntryCarZone(nZone));
  }

  function f_OutCar(numCtrl nCtrl,numZone nZone) runs on System
  {
      cEnv_pTesting.send(a_sOutCarCtrl(nCtrl));
      cEnv_pTesting.send(a_sOutCarZone(nZone));
  }

  function f_RequestInfoCtrl_Zone(numCtrl nCtrl,numZone nZone) runs on System
  {
      cDisplay_Main.send(a_sReqInfoCtrlZone(nCtrl,nZone));
  }

  /*TestCases*/
  testcase tc_EntryCar(numCtrl p_numCtrl,numZone p_numZone) runs on System
  {
      f_EntryCar(p_numCtrl,p_numZone);
      setverdict(pass);
  }
  testcase tc_OutCar(numCtrl p_numCtrl,numZone p_numZone) runs on System
  {
      f_OutCar(p_numCtrl,p_numZone);
      setverdict(pass);
  }

  testcase tc_CreationCtrl() runs on System
  {
      cEnv_Main.send(a_sCreateCtrlZone);
      alt
      {
        []
        cEnv_Main.receive(a_sOkCreateCtrl)
        {
          setverdict(pass);
        }
        [else]
        {
          setverdict(fail);
        }
      }
      stop;
  }

   testcase tc_CreationZone(numCtrl p_numCtrl,spots p_totalSpots,spots p_freeSpots) r
 uns on System
   {
      cEnv_Main.send(a_sAddZone(p_numCtrl,p_totalSpots,p_freeSpots));
      alt
      {
        []
        cEnv_Main.receive(a_sOkCreationZone)
        {
          setverdict(pass);
        }
        [else]
        {
          setverdict(fail);
        }
      }
      stop;
   }

   testcase tc_VerifyFreeSpots(numCtrl nCtrl_Req,numZone nZone_Req,spots freeSpots) r
 uns on System
   {
      f_RequestInfoCtrl_Zone(nCtrl_Req,nZone_Req);
      alt
      {
          []
          cDisplay_Main.receive(a_sInfoCtrlZone(freeSpots))
          {
            setverdict(pass);
          }
          [else]
          {
            setverdict(fail);
          }
      }
      stop;
   }
   testcase tc_initialization() runs on System
   {
      timer t_WaitInitializationSystem;
      t_WaitInitializationSystem.start(1);
      t_WaitInitializationSystem.timeout;
      setverdict(pass);

   }
```

Continue...

```
control
{
  var integer index,count_numCars;
  var integer nCtrl_Entry;
  var integer nZone_Entry;
  var integer numCars;
  var integer indexHour; /*Each Hour has a index assigned */
  timer t_waitEntryCar;

  /*TestCase to Initilization of whole the system */

  execute(tc_initialization());

  /* TestCase to creation of pCtrl and pZone */

  execute(tc_CreationCtrl());
  execute(tc_CreationZone(0,300,300));
  execute(tc_CreationZone(1,300,300));
  execute(tc_CreationZone(1,300,300));

  /*Loop for In and Out of Cars to the Parking System*/


  indexHour:=19;

  for (index:=0;index<4*indexHour+4;index:=index+1)
  {
    /* In cars */

    numCars:=aEntryCar[index];
    nCtrl_Entry:=aCtrlEntryCar[index];
    nZone_Entry:=aZoneEntryCar[index];
    for (count_numCars:=0; count_numCars<numCars;count_numCars:=count_numCars+1)
    {
        execute(tc_EntryCar(nCtrl_Entry,nZone_Entry));
        t_waitEntryCar.start(8);
        t_waitEntryCar.timeout;
    }
    /* Out Cars */
    numCars:=aOutCar[index];
    nCtrl_Entry:=aCtrlOutCar[index];
    nZone_Entry:=aZoneOutCar[index];
    for (count_numCars:=0; count_numCars<numCars;count_numCars:=count_numCars+1)
    {
        execute(tc_OutCar(nCtrl_Entry,nZone_Entry));
        t_waitEntryCar.start(8);
        t_waitEntryCar.timeout;
    }


  }
  numCars:=aExpectedSpots[4*indexHour];
  nCtrl_Entry:=aCtrlExpected[4*indexHour];
  nZone_Entry:=aZoneExpected[4*indexHour];
  execute(tc_VerifyFreeSpots(nCtrl_Entry,nZone_Entry,numCars));


    }
 }
```