```
// Standard library for the PCGSL programming language

// ******************************
// Functions for string conversion
// ******************************

//
// stringtoint - returns the int representation of the given string, within the
//               given integer limits. Or null, if no int representation found.
//
// input type(s) expected: string, int, int
// output type(s): int
//
stringtoint(var s, var low, var high) {
  var j;

  if (@s != "string" || @low != "int" || @high != "int") {
    return null;
  }

  j = low;

  while (true) {
    if (j > high) {
      return null;
    }
    if ("" ^ j == s) {
      return j;
    }
    j++;
  }
}

//
// stringtocard - returns the Card representation of the given string. Or null
//                if no Card is found.
//
// input type(s) expected: string
// output type(s): Card
//
stringtocard(var s) {
  var cards;

  if (@s != "string") {
    return null;
  }

  cards = listallcards();

  j = 0;

  while (true) {
    if (j >= |cards|) {
      return null;
    }
    if ("" ^ cards[j] == s) {
      return cards[j];
    }
    j++;
  }
}

// ******************
// Functions for lists
// ******************

//
// listlength - returns the length of the given list.
//
```

```
// input type(s) expected: list
// output type(s): int
//
listlength(var l) {
  if (@l != "list") {
    return null;
  }

  return |l|;
}

//
// listfind - returns the index of the element in the given list matching the
//            given input item, or null if none exists. If more than one match,
//            returns the first occurring match in the list.
//
// input type(s) expected: list, var
// output type(s): int
//
listfind(var l, var e) {
  var i;

  if (@l != "list") {
    return null;
  }

  i = 0;
  while (true) {
    if (i == |l|) {
      return null;
    }
    if (l[i] == e) {
      return i;
    }
    i++;
  }
}

//
// listremove - returns a new list that has removed the given index's element
//              from the given list. If the index is out of bounds, returns
//              a new list identical to the given list.
//
// input type(s) expected: list, int
// output type(s): list
//
listremove(var oldl, var i) {
  var newl;
  var j;

  if (@oldl != "list" || @i != "int") {
    return null;
  }

  if (i < 0 || i >= |oldl|) {
    return oldl;
  }

  newl = [];
  j = 0;

  while (true) {
    if (j >= |oldl|) {
      return newl;
    }
    if (j != i) {
      newl = newl :: oldl[j];
    }
    j++;
```

```
  }
}

//
// listreverse - returns a new list that flips the elements of the given list.
//
// input type(s) expected: list
// output type(s): list
//
listreverse(var oldl) {
  var newl;
  var i;

  if (@oldl != "list") {
    return null;
  }

  newl = [];
  i = |oldl| - 1;

  while (true) {
    if (i < 0) {
      return newl;
    }
    newl = newl :: oldl[i];
    i--;
  }
}

// ************************************
// Functions for Cards and CardEntities
// ************************************

//
// cardsuit - returns the suit value of the given Card as an int. Heart is 1,
//            Diamond is 2, Club is 3, and Spade is 4.
//
// input type(s) expected: Card
// output type(s): int
//
cardsuit(var c) {
  if (@c != "Card") {
    return null;
  }

  if (c == H2 || c == H3 || c == H4 || c == H5 || c == H6 ||
      c == H7 || c == H8 || c == H9 || c == H10 || c == HJ ||
      c == HQ || c == HK || c == HA) {
    return 1;
  }

  if (c == D2 || c == D3 || c == D4 || c == D5 || c == D6 ||
      c == D7 || c == D8 || c == D9 || c == D10 || c == DJ ||
      c == DQ || c == DK || c == DA) {
    return 2;
  }

  if (c == C2 || c == C3 || c == C4 || c == C5 || c == C6 ||
      c == C7 || c == C8 || c == C9 || c == C10 || c == CJ ||
      c == CQ || c == CK || c == CA) {
    return 3;
  }

  if (c == S2 || c == S3 || c == S4 || c == S5 || c == S6 ||
      c == S7 || c == S8 || c == S9 || c == S10 || c == SJ ||
      c == SQ || c == SK || c == SA) {
    return 4;
  }
}
```

```
//
// cardface - returns the face value of the given Card as an int. Ace is 1.
//
// input type(s) expected: Card
// output type(s): int
//
cardface(var c) {
  if (@c != "Card") {
    return null;
  }

  if (c == H2 || c == D2 || c == C2 || c == S2) {
    return 2;
  }

  if (c == H3 || c == D3 || c == C3 || c == S3) {
    return 3;
  }

  if (c == H4 || c == D4 || c == C4 || c == S4) {
    return 4;
  }

  if (c == H5 || c == D5 || c == C5 || c == S5) {
    return 5;
  }

  if (c == H6 || c == D6 || c == C6 || c == S6) {
    return 6;
  }

  if (c == H7 || c == D7 || c == C7 || c == S7) {
    return 7;
  }

  if (c == H8 || c == D8 || c == C8 || c == S8) {
    return 8;
  }

  if (c == H9 || c == D9 || c == C9 || c == S9) {
    return 9;
  }

  if (c == H10 || c == D10 || c == C10 || c == S10) {
    return 10;
  }

  if (c == HJ || c == DJ || c == CJ || c == SJ) {
    return 11;
  }

  if (c == HQ || c == DQ || c == CQ || c == SQ) {
    return 12;
  }

  if (c == HK || c == DK || c == CK || c == SK) {
    return 13;
  }

  if (c == HA || c == DA || c == CA || c == SA) {
    return 14;
  }
}

//
// listallcards - returns a list containing all Cards.
//
// input type(s) expected:
```

```
// output type(s): list of cards
//
listallcards() {
  return [H2, H3, H4, H5, H6, H7, H8, H9, H10, HJ, HQ, HK, HA,
          D2, D3, D4, D5, D6, D7, D8, D9, D10, DJ, DQ, DK, DA,
          C2, C3, C4, C5, C6, C7, C8, C9, C10, CJ, CQ, CK, CA,
          S2, S3, S4, S5, S6, S7, S8, S9, S10, SJ, SQ, SK, SA];
}

//
// containscard - returns true if the given CardEntity contains the given Card,
//                and false otherwise.
//
// input type(s) expected: CardEntity, Card
// output type(s): boolean
//
containscard(var e, var c) {
  var i;

  if (@e != "CardEntity" || @c != "Card") {
    return null;
  }

  i = 0;

  while (true) {
    if (i >= |e|) {
      return false;
    }
    if (e[i] == c) {
      return true;
    }
    i++;
  }
}

//
// locatecard - returns the CardEntity, from a given list, that contains the
//              given Card. Returns null if no such CardEntity is found.
//
// input type(s) expected: list of CardEntities, Card
// output type(s): CardEntity
//
locatecard(var entities, var c) {
  var i;

  if (@entities != "list" || @c != "Card") {
    return null;
  }

  i = 0;

  while (true) {
    if (i >= |entities|) {
        return null;
    }
    if (@entities[i] == "CardEntity") {
      if (containscard(entities[i], c)) {
        return entities[i];
      }
    }
    i++;
  }
}

//
// shuffle - randomly reorders the Cards in the given CardEntity.
//
// input type(s) expected: CardEntity
```

```
// output type(s): null
//
shuffle(var e) {
  var r;
  var l;
  var i;
  var j;

  if (@e != "CardEntity") {
    return null;
  }

  l = [];
  i = 0;
  j = 0;

  while (i < |e|) {
    l = l :: e[i];
    i++;
  }

  while (true) {
    if (i <= 0 || j >= |e|) {
      return null;
    }
    r = ~i;
    e <- l[r];
    l = listremove(l, r);
    i--;
    j++;
  }
}

//
// transferall - transfers all Cards to the first given CardEntity from the
//               second given CardEntity.
//
// input type(s) expected: CardEntity, CardEntity
// output type(s): null
//
transferall(var e1, var e2) {
  if (@e1 != "CardEntity" || @e2 != "CardEntity" || e1 == e2) {
    return null;
  }

  while (|e2| > 0) {
    e1 <- e2[0];
  }
}
```