The materials included in these files are intended for use by AP teachers for course and exam preparation in the classroom; permission for any other use must be sought from the Advanced Placement Program®. Teachers may reproduce them, in whole or in part, in limited quantities, for face-to-face teaching purposes but may not mass distribute the materials, electronically or otherwise. These materials and any copies made of them may not be resold, and the copyright notices must be retained as they appear here. This permission does not apply to any third-party copyrights contained herein.
(a) You will write the Position member function Northeast, which is described as follows. Northeast should return the position in the environment to the northeast of the current position. In the diagram shown above, if pos1 is the position (2, 1), the call pos1.Northeast() returns the position (1, 2), and if pos2 is the position (2, 9), the call pos2.Northeast() returns the position (1, 10).

Complete function Northeast below.

```cpp
Position Position::Northeast() const

// postcondition: returns Position northeast of this position

2     return Position(myRow-1, myCol+1);
3```

GO ON TO THE NEXT PAGE.
(b) You will write the Fish member function ForwardNbrs, which is described as follows.
ForwardNbrs should return a neighborhood consisting of those positions that meet the requirements for Potential Movement Locations.

In writing ForwardNbrs, you may use any of the Fish and Position member functions. Assume that these functions, including Position::Northeast, work as specified, regardless of what you wrote in part (a).

An implementation of this function distinguishes among multiple cases based on direction. In writing your code, you must show the code for the two specific cases, north and northeast. You may write "..." to indicate where the remaining cases should be. All statements other than these remaining cases must be shown.

Complete function ForwardNbrs below.

Neighborhood Fish::ForwardNbrs(const Environment & env) const
// postcondition: returns empty neighbors in a forward direction from
// myDir - straight ahead and diagonally ahead to the
// right or left
{
    Neighborhood nbrs;
    if (myDir == "N")
    {
        AddIfEmpty(env, nbrs, myPos.North());
        AddIfEmpty(env, nbrs, myPos.Northwest());
        AddIfEmpty(env, nbrs, myPos.Northeast());
    }
    if (myDir == "NE")
    {
        AddIfEmpty(env, nbrs, myPos.Northeast());
        AddIfEmpty(env, nbrs, myPos.North());
        AddIfEmpty(env, nbrs, myPos.East());
    }

    ...

    return nbrs;
}
(c) You will write the Position member function DirectionTo, which returns the direction from this Position to Position other.

An implementation of this function distinguishes among multiple cases based on direction. In writing your code, you must show the code for the two specific cases, north and northeast. You may write "..." to indicate where the remaining cases should be. All statements other than these remaining cases must be shown.

Complete function DirectionTo below.

```cpp
apstring Position::DirectionTo(const Position & other) const
// precondition: other is adjacent to this Position
// postcondition: returns the string representation of the direction
// from this Position to other
{
    apstring direct;
    if ( North() == other )
        direct = "N";
    if ( Northeast() == other )
        direct = "NE";
    ...
    return direct;
}
```

GO ON TO THE NEXT PAGE.
(a) You will write the Position member function Northeast, which is described as follows. Northeast should return the position in the environment to the northeast of the current position. In the diagram shown above, if pos1 is the position (2, 1), the call pos1.Northeast() returns the position (1, 2), and if pos2 is the position (2, 9), the call pos2.Northeast() returns the position (1, 10).

Complete function Northeast below.

```cpp
Position Position::Northeast() const
// postcondition: returns Position northeast of this position
```

```cpp
Position Position::Northeast() const
{
    // return Position (myRow - 1, myCol + 1);
} 
```

GO ON TO THE NEXT PAGE.
(b) You will write the Fish member function ForwardNbrs, which is described as follows. ForwardNbrs should return a neighborhood consisting of those positions that meet the requirements for Potential Movement Locations.

In writing ForwardNbrs, you may use any of the Fish and Position member functions. Assume that these functions, including Position::Northeast, work as specified, regardless of what you wrote in part (a).

An implementation of this function distinguishes among multiple cases based on direction. In writing your code, you must show the code for the two specific cases, north and northeast. You may write "..." to indicate where the remaining cases should be. All statements other than these remaining cases must be shown.

Complete function ForwardNbrs below.

Neighborhood Fish::ForwardNbrs(const Environment & env) const
// postcondition: returns empty neighbors in a forward direction from
// myDir - straight ahead and diagonally ahead to the
// right or left

Neighborhood Nbrs;
if (myDir == "N")
  AddIfEmpty (env, nbrs, myPos, North ());
  AddIfEmpty (env, nbrs, myPos, Northeast ());
  AddIfEmpty (env, nbrs, myPos, Northwest ());
  return nbrs;

if (myDir == "NE")
  AddIfEmpty (env, nbrs, myPos, Northeast ());
  AddIfEmpty (env, nbrs, myPos, North ());
  AddIfEmpty (env, nbrs, myPos, East ());
  return nbrs;

... // other cases of myDir

GO ON TO THE NEXT PAGE.
(c) You will write the Position member function DirectionTo, which returns the direction from this Position to Position other.

An implementation of this function distinguishes among multiple cases based on direction. In writing your code, you must show the code for the two specific cases, north and northeast. You may write "..." to indicate where the remaining cases should be. All statements other than these remaining cases must be shown.

Complete function DirectionTo below.

```cpp
// precondition: other is adjacent to this Position
// postcondition: returns the string representation of the direction from this Position to other

string Position::DirectionTo(const Position & other) const {
    if (myCol == other.Col() &&
        myRow == other.Row() + 1)
        return "N";
    if (myCol == other.Col() - 1 &&
        myRow == other.Row() + 1)
        return "NE";
    // other directions
}
```

GO ON TO THE NEXT PAGE.
(a) You will write the Position member function Northeast, which is described as follows. Northeast should return the position in the environment to the northeast of the current position. In the diagram shown above, if pos1 is the position (2, 1), the call pos1.Northeast() returns the position (1, 2), and if pos2 is the position (2, 9), the call pos2.Northeast() returns the position (1, 10).

Complete function Northeast below.

```cpp
Position Position::Northeast() const
// postcondition: returns Position northeast of this position
{
    return Position(myRow + 1, myCol + 1);
}
```

GO ON TO THE NEXT PAGE.
(b) You will write the Fish member function ForwardNbrs, which is described as follows. ForwardNbrs should return a neighborhood consisting of those positions that meet the requirements for Potential Movement Locations.

In writing ForwardNbrs, you may use any of the Fish and Position member functions. Assume that these functions, including Position::Northeast, work as specified, regardless of what you wrote in part (a).

An implementation of this function distinguishes among multiple cases based on direction. In writing your code, you must show the code for the two specific cases, north and northeast. You may write "..." to indicate where the remaining cases should be. All statements other than these remaining cases must be shown.

Complete function ForwardNbrs below.

```cpp
Neighborhood Fish::ForwardNbrs(const Environment & env) const {
    // postcondition: returns empty neighbors in a forward direction from
    // myDir - straight ahead and diagonally ahead to the
    // right or left

    Neighborhood nbrs;

    if (myDir == "N") {
        AddIfEmpty(env, nbrs, pos, North());
        AddIfEmpty(env, nbrs, pos, Northeast());
    }

    if (myDir == "NE") {
        AddIfEmpty(env, nbrs, pos, North());
        AddIfEmpty(env, nbrs, pos, Northeast());
        AddIfEmpty(env, nbrs, pos, East());
    }

    return nbrs;
}
```

GO ON TO THE NEXT PAGE.
(c) You will write the Position member function DirectionTo, which returns the direction from this Position to Position other.

An implementation of this function distinguishes among multiple cases based on direction. In writing your code, you must show the code for the two specific cases, north and northeast. You may write "..." to indicate where the remaining cases should be. All statements other than these remaining cases must be shown.

Complete function DirectionTo below.

```cpp
string Position::DirectionTo(const Position & other) const
// precondition: other is adjacent to this Position
// postcondition: returns the string representation of the direction from this Position to other
```

```cpp
int row1, col1;
int temp1, temp2;
row1 = other.row();
col1 = other.col();

if ((row1 - row) == 1 && (col1 - col) == 1)
    return ("Northeast");

if ((row1 - row) == 0 && (col1 - col) == 1)
    return ("North");
```

...