```
////////////////////////////////////////////////////////////////////////////////
//
//  COMS20600 - WEEK 1 / LAB WARM-UP (OPTIONAL EXERCISE)
//              LAB SAMPLE SOLUTION
//              TITLE: "EMBARRASSINGLY PARALLEL ANTS"
//
//  OBJECTIVE:  exercise in using XDE and basic XC language constructs:
//              parallel threads, multiple function returns and array access
//
//  NARRATIVE:  Imagine a (fairly primitive) ant species where individuals
//              are completely ignorant to conspecifics. The species lives
//              in a 3x4 (2D) constant array world that is a closed domain
//              (e.g. leaving the array at the south edge leads to the north
//              edge again etc). Each entry in the array is an unsigned char
//              that represents the fertility of this location:
//
//                 10   0   1   7
//                  2   10  0   3
//                  6   8   7   6
//
//              Ants have a position and a food counter; two or more ants
//              can be at the same position at one time. Ants are always
//              on the move. They move either east or south, preferring the
//              location that is more fertile. When entering a new location,
//              an ant's food counter is increased by the fertility value of
//              the region entered.
//
//              Implement a small XC program which…
//              -   defines a constant array world as given above
//              -   initialises the position of four ants at the four least fertile locations
//              -   hands the world, an ant id and start position as  parameters to a function ant
//              -   runs 4 such ant function calls representing 4 ants in an 'embarrassingly parallel'
//                  fashion, each moving 2 times printing the move info
//              -   each ant checks all food items after a move, printing a check info
//              -   each function call reports back both the food counter and the final position
//                  coordinates of the ant (multi-value return)
//              -   once all ants have moved 2 steps, the program prints the overall food gathered
//                  between the four ants and the mean position of the four ants after their foraging walks
//
////////////////////////////////////////////////////////////////////////////////
```

```
#include <stdio.h>

//PROCESS representing an independent ant
{int,int,int} ant (  unsigned int id,                //the ant identifier
                     const unsigned char w[3][4],    //the constant world array
                     unsigned int x,                 //the starting position x
                     unsigned int y                  //the starting position y
                   ) {

  unsigned int food = 0;  //food counter of the ant

  //print start information
  printf("Ant %d starting...\n", id);

  //MOVE TWO ITERATIONS
  for(int i=0; i<2; i++) {
    //check land fertility in east and south
    if ( w[(x+1)%3][y] > w[x][(y+1)%4] )
      //move east
      x = (x+1)%3;
    else
      //move south
      y = (y+1)%4;

    //increase food counter by current land fertility
    food += w[x][y];

    //announce move
    printf("Ant %d moved to (%d,%d) with new food count %d\n", id, x, y, food);

    //announce food item checks
    for(int j=0; j<food; j++)
      printf("Ant %d checking food item %d/%d\n", id, j, food);
  }

  //announce end of work
  printf("Ant %d finishes work at position (%d,%d) with food count %d\n", id, x, y, food);

  //report back food counter and position
  return {food, x, y};
}
```

```
//MAIN PROCESS initialising variables, sparking four ant processes & combining results
int main ( void ) {

    //1. INIT VARIABLES (AVOID GLOBAL, SHARED VARIABLES!)
    const unsigned char world[3][4] = {{10,0,1,7},{2,10,0,3},{6,8,7,6}};  //the world
    unsigned int  food[4];       //food reported as harvested per ant
    unsigned int  x[4], y[4];    //end positions reported per ant
    unsigned int  allFood = 0;   //final overall food count
    unsigned int  sumX = 0;
    unsigned int  sumY = 0;

    //2. RUN FOUR ANT PROCESSES IN PARALLEL
    //parallel execution of four different ant processes, which report back food and position
    par {
      {food[0],x[0],y[0]} = ant(0,world,0,1); //run concurrent ant process 1
      {food[1],x[1],y[1]} = ant(1,world,1,2); //run concurrent ant process 2
      {food[2],x[2],y[2]} = ant(2,world,0,2); //run concurrent ant process 3
      {food[3],x[3],y[3]} = ant(3,world,1,0); //run concurrent ant process 4
    } // <-- WAIT HERE UNTIL ALL ANTS HAVE REPORTED RESULTS

    //3. REDUCTION STEP
    //summation of food gathered by the four ants and final positions
    for(int i=0; i<4; i++) {
      allFood += food[i];
      sumX += x[i];
      sumY += y[i];
    }

    //4. REPORT RESULTS OF HARVEST
    printf("Food: %d   avg-X: %d/10   avg-Y: %d/10\n", allFood, 10*sumX/4, 10*sumY/4);

    //DONE & TERMINATE PROGRAM
    return 0;
}
```