```
    1: #include <algorithm>
    2: #include <iostream>
    3: #include <mpi.h>
    4: #include <vector>
    5: using namespace std;
    6:
    7:
    8: void DebugVector(const vector<double> &xin, MPI_Comm icomm) {
    9:     int rank, size;
   10:     MPI_Comm_rank(icomm, &rank);
   11:     MPI_Comm_size(icomm, &size);
   12:
   13:     int next_process = 0;
   14:     while (next_process != -1) {
   15:         // Print the local vector for each process
   16:         if (rank==next_process){
   17:             cout << "x_" << rank << " = ";
   18:             for (const auto& value : xin) {
   19:                 cout << value << " ";
   20:             }
   21:             cout << endl;
   22:         }
   23:         MPI_Barrier(icomm);
   24:
   25:         if (rank == 0) {
   26:             cout << "Enter rank (0-" << size - 1 << ") or -1 to exit: ";
   27:             cin >> next_process;
   28:         }
   29:         MPI_Bcast(&next_process, 1, MPI_INT, 0, icomm);
   30:         MPI_Barrier(icomm);
   31:     }
   32: }
   33:
   34:
   35:
   36: double par_scalar(const vector<double>& x, const vector<double>& y, MPI_Comm icomm)
{
   37:
   38:     double local_dot = 0.0;
   39:     for (size_t i = 0; i < x.size(); ++i) {
   40:         local_dot += x[i] * y[i];
   41:     }
   42:
   43:     double global_dot = 0.0;
   44:     MPI_Allreduce(&local_dot, &global_dot, 1, MPI_DOUBLE, MPI_SUM, icomm);
   45:
   46:     return global_dot;
   47: }
   48:
   49:
   50: tuple<double,double> find_global_minmax(const vector<double>& xin, MPI_Comm icomm) {
   51:     int rank, size;
   52:     MPI_Comm_rank(icomm, &rank);
   53:     MPI_Comm_size(icomm, &size);
   54:     // Find local min/max
   55:     double local_min = *min_element(xin.begin(), xin.end());
   56:     double local_max = *max_element(xin.begin(), xin.end());
   57:     // Gather local mins/maxs in vector
   58:     vector<double> local_min_vector(size);
   59:     vector<double> local_max_vector(size);
   60:     MPI_Gather(&local_min, 1, MPI_DOUBLE, local_min_vector.data(), 1, MPI_DOUBLE, 0,
 icomm);
   61:     MPI_Gather(&local_max, 1, MPI_DOUBLE, local_max_vector.data(), 1, MPI_DOUBLE, 0,
 icomm);
   62:     // Find global min/max
   63:     double global_min(0);
   64:     double global_max(0);
   65:     if (rank==0) {
```

```
    66:         global_min = *min_element(local_min_vector.begin(), local_min_vector.end());
    67:         global_max = *max_element(local_max_vector.begin(), local_max_vector.end());
    68:     }
    69:     // Broadcast global min/max
    70:     MPI_Bcast(&global_min, 1, MPI_DOUBLE, 0, icomm);
    71:     MPI_Bcast(&global_max, 1, MPI_DOUBLE, 0, icomm);
    72:
    73:     return make_tuple(global_min, global_max);
    74: }
    75:
    76:
    77: tuple<double,double> find_global_minmax_Allreduce(const vector<double>& xin, MPI_Com
m icomm) {
    78:     double local_min = *min_element(xin.begin(), xin.end());
    79:     double local_max = *max_element(xin.begin(), xin.end());
    80:     double global_min(0);
    81:     double global_max(0);
    82:     MPI_Allreduce(&local_min, &global_min, 1, MPI_DOUBLE, MPI_MIN, icomm);
    83:     MPI_Allreduce(&local_max, &global_max, 1, MPI_DOUBLE, MPI_MAX, icomm);
    84:     return make_tuple(global_min, global_max);
    85: }
    86:
    87:
    88: int main(int argc, char *argv[]) {
    89:     MPI_Comm icomm = MPI_COMM_WORLD;
    90:     MPI_Init(&argc, &argv);
    91:     int rank, size;
    92:     MPI_Comm_rank(icomm, &rank);
    93:     MPI_Comm_size(icomm, &size);
    94:
    95:     if (rank==0) {
    96:         cout << "\n There are " << size << " processes running.\n";
    97:     }
    98:
    99:     // Create vectors
   100:     size_t n=20;
   101:     vector<double> local_vector(n);
   102:     vector<double> local_vector_inv(n);
   103:     for (size_t i=0; i<n; ++i) {
   104:         // local_vector[i] = rank*n + i+1;
   105:         // local_vector_inv[i] = 1.0/(local_vector[i]);
   106:
   107:         local_vector[i] = rank*100.0 + (i%5)*10.0 + i;      // EX8
   108:         local_vector_inv[i] = 1.0/(local_vector[i]+1.0);
   109:     }
   110:
   111:
   112:     MPI_Barrier(icomm);
   113:     if (rank == 0) {printf("\n\n-------------- Task 5 --------------\n\n");}
   114:     DebugVector(local_vector, icomm);
   115:
   116:
   117:     MPI_Barrier(icomm);
   118:     if (rank == 0) {printf("\n\n-------------- Task 6 --------------\n\n");}
   119:     double result = par_scalar(local_vector, local_vector_inv, icomm);
   120:     if (rank == 0) {printf("Global scalar product: %f\n", result);}
   121:
   122:
   123:     MPI_Barrier(icomm);
   124:     if (rank == 0) {printf("\n\n-------------- Task 7 --------------\n\n");}
   125:     auto [min, max] = find_global_minmax(local_vector, icomm);
   126:     if (rank == 0) {printf("Global min: %.0f | global max: %.0f\n\n", min, max);}
   127:
   128:     MPI_Barrier(icomm);
   129:     tuple(min, max) = find_global_minmax_Allreduce(local_vector, icomm);
   130:     if (rank == 0) {printf("Global min: %.0f | global max: %.0f\n", min, max);}
   131:
   132:
```

Exchange !?

```
  133:     MPI_Barrier(icomm);
  134:     if (rank == 0) {printf("\n\n-------------- Task 8 --------------\n\n");}
  135:
  136:     if (rank == 0) {printf("\n---- MPI_Alltoall ----\n");}
  137:     vector<double> recv(n);
  138:     MPI_Alltoall(local_vector.data(), 5, MPI_DOUBLE, recv.data(), 5, MPI_DOUBLE, ico
mm);
  139:     DebugVector(recv, icomm);
  140:
  141:     MPI_Barrier(icomm);
  142:     if (rank == 0) {printf("\n---- MPI_Alltoall using MPI_IN_PLACE ----\n");}
  143:     MPI_Alltoall(MPI_IN_PLACE, 0, MPI_DATATYPE_NULL, local_vector.data(), 5, MPI_DOU
BLE, icomm);
  144:     DebugVector(local_vector, icomm);
  145:
  146:
  147:     MPI_Finalize();
  148:     return 0;
  149: }
```