Lab Report

Submitted by: - Parth Parashar

Summary

This was a helpful lab assignment for me as I got to learn a lot about the use of domains.

For the first question, It was fairly easy to judge how the programs were working and what the functionalities of the domains were. I got to learn about the behaviour of domains by making changes to the programs and the mapping of domains.

The outputs and the changed code for those outputs is given in this lab report below.

For the second question and third questions, since they were based on domains as well (including domain maps), I got a deeper understanding of the topic.

For the fourth question which was on matrix multiplication, for converting it to PGAS version was a little challenging. The use of domain maps in the third question gave me an idea on how to convert it to PGAS version. I used domain maps to map domains to the different variables and got the expected output.

After introducing the verification code to the matrix multiplication problem, I was able to correctly verify the output. The domain maps are working properly.

As for the Gauss-Seidel problem, I was stuck on the problem for quite some time. I was not able to figure out how to converge the values which were being updated.

The first thing I did was, I used a temporary variable to store the values in the for loop and just like Jacobi, I used the delta and epsilon values to calculate the requisite values (as well as loop ending condition).

The expected value still was not correct, but it is not being stuck in the while loop which was the case when I had not included the delta and epsilon conditions and was using a while(true) statement.

The second thing which I tried was removed the temporary variable and then ran it again without the reduction condition and used only the same variable changes. The answer still did not converge. The code and outputs are attached in the zip file as well as in this lab document as well.

For the File-I/O problem, I made the changes to the declaration of N as well as integer values and got the correct output after a couple of tries. It took me quite some time to get to the correct output and it was a good introduction to the File IO operations.

For the optional part, I tried to create a worker routine to run the program but was unable to successfully integrate the functionality as required and the output is not as expected.

The code for the same is attached in the zip file.

Question-1: -

```
parth2@ada:~/Desktop/parallelProgramming/lab9$ ./domain1 --numLocales=2
D = {1..6}       // a = [i in D] i;
a = 1 2 3 4 5 6 // a has 6 elements

D = {2..4}       // redefine domain D
a = 2 3 4        // a has 3 elements now

D = {4..8}       // redefine domain D again
a = 4 0 0 0 0    // a has 5 elements now
```

When we change the domain to D = {1..2}

OpenSSH SSH client

```
parth2@ada:~/Desktop/parallelProgramming/lab9$ cat domain1.chpl
//------------------------------------------------------------------------
// Program code for CS 415/515 Parallel Programming, Portland State University.
//------------------------------------------------------------------------

// Domain and array examples.
//

var D = {1..2};          // declare a domain
var a: [D] int;          // declare an int array over domain
a = [i in D] i;          // a = [1,2,3,4,5,6]

writeln("D = ", D, "\t// a = [i in D] i;");
writeln("a = ", a, "\t// a has ", a.size, " elements\n");

// redefine domain D, what happens to a?
D = {2..4};
writeln("D = ", D, "\t// redefine domain D");
writeln("a = ", a, "\t// a has ", a.size, " elements now\n");

// redefine domain D, what happens to a?
D = {4..8};
writeln("D = ", D, "\t// redefine domain D again");
writeln("a = ", a, "\t// a has ", a.size, " elements now");

parth2@ada:~/Desktop/parallelProgramming/lab9$
```

Then the output changes to

```
parth2@ada:~/Desktop/parallelProgramming/lab9$ ./domain1 --numLocales=2
D = {1..2}       // a = [i in D] i;
a = 1 2 // a has 2 elements

D = {2..4}       // redefine domain D
a = 2 0 0        // a has 3 elements now

D = {4..8}       // redefine domain D again
a = 0 0 0 0 0    // a has 5 elements now
parth2@ada:~/Desktop/parallelProgramming/lab9$
```

Question-2: -

```
parth2@ada:~/Desktop/parallelProgramming/lab9$ ./domain2 --numLocales=5
D = {1..6 by 2} // a striding domain
a = 1 3 5       // a is defined over D; a has 3 elements

D = {1..8 by 2} // redefine D to a superset of original D
a = 1 3 5 0     // a has 4 elements now

D = {1..8}      // redefine D to another superset
a = 1 0 3 0 5 0 0 0     // a has 8 elements now

D = {2..7 by 2} // one more, a disjoint domain
a = 0 0 0       // a has 3 elements now
```

When changing the domain, we can see that

OpenSSH SSH client

```
parth2@ada:~/Desktop/parallelProgramming/lab9$ cat domain2.chpl
//-----------------------------------------------------------------------
// Program code for CS 415/515 Parallel Programming, Portland State University.
//-----------------------------------------------------------------------

// More domain and array examples.
//

var D = {1..10} by 2;          // declare a domain D
var a: [D] int = [i in D] i; // declare an array over D

writeln("D = ", D, "\t// a striding domain");
writeln("a = ", a, "\t// a is defined over D; a has ", a.size, " elements\n");

// redefine domain D, what happens to a?
D = {1..8} by 2;
writeln("D = ", D, "\t// redefine D to a superset of original D");
writeln("a = ", a, "\t// a has ", a.size, " elements now\n");

// redefine domain D, what happens to a?
D = {1..8} by 1;
writeln("D = ", D, "\t// redefine D to another superset");
writeln("a = ", a, "\t// a has ", a.size, " elements now\n");

// redefine domain D, what happens to a?
D = {2..7} by 2;
writeln("D = ", D, "\t// one more, a disjoint domain");
writeln("a = ", a, "\t// a has ", a.size, " elements now");

parth2@ada:~/Desktop/parallelProgramming/lab9$
```

The output changes to: -

```
parth2@ada:~/Desktop/parallelProgramming/lab9$ chpl -g -o domain2 domain2.chpl
parth2@ada:~/Desktop/parallelProgramming/lab9$ ./domain2 --numLocales=5
D = {1..10 by 2}        // a striding domain
a = 1 3 5 7 9   // a is defined over D; a has 5 elements

D = {1..8 by 2} // redefine D to a superset of original D
a = 1 3 5 7     // a has 4 elements now

D = {1..8}      // redefine D to another superset
a = 1 0 3 0 5 0 7 0     // a has 8 elements now

D = {2..7 by 2} // one more, a disjoint domain
a = 0 0 0       // a has 3 elements now
parth2@ada:~/Desktop/parallelProgramming/lab9$
```

Question-3: -

```
parth2@ada:~/Desktop/parallelProgramming/lab9$ chpl -g -o domainmap domainmap.chpl
parth2@ada:~/Desktop/parallelProgramming/lab9$ ./domainmap --numLocales=5
a:  0 0 0 0 0 0 0 0 0
b1: 0 0 1 1 2 3 3 4
b2: 0 0 1 2 3 4 4 4
b3: 0 0 0 1 1 2 2 2
parth2@ada:~/Desktop/parallelProgramming/lab9$
```

Now, When I change the mapping,

OpenSSH SSH client

```
parth2@ada:~/Desktop/parallelProgramming/lab9$ cat domainmap.chpl
//------------------------------------------------------------------------
// Program code for CS 415/515 Parallel Programming, Portland State University.
//------------------------------------------------------------------------

// Domain Map Example
//
// linux> ./domainmap -nl 4
//

use BlockDist;

const D: domain(1) = {1..16};
const D1 = D dmapped Block(D);
const D2 = D dmapped Block({2..6});
const D3 = D dmapped Block({1..12});

var a:  [D] int;
var b1: [D1] int;
var b2: [D2] int;
var b3: [D3] int;

forall e in a do e = here.id;
forall e in b1 do e = here.id;
forall e in b2 do e = here.id;
forall e in b3 do e = here.id;

writeln("a:  ", a);
writeln("b1: ", b1);
writeln("b2: ", b2);
writeln("b3: ", b3);
```

then we can see the following output.

OpenSSH SSH client

```
parth2@ada:~/Desktop/parallelProgramming/lab9$ ./domainmap --numLocales=5
a:  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
b1: 0 0 0 0 1 1 1 2 2 2 3 3 3 4 4 4
b2: 0 0 1 2 3 4 4 4 4 4 4 4 4 4 4 4
b3: 0 0 0 1 1 2 2 2 3 3 4 4 4 4 4 4
parth2@ada:~/Desktop/parallelProgramming/lab9$
```

When we change the domain to 1..16, then we can see the output changed to: -

OpenSSH SSH client

```
parth2@ada:~/Desktop/parallelProgramming/lab9$ vi domainmap.chpl
parth2@ada:~/Desktop/parallelProgramming/lab9$ chpl -g -o domainmap domainmap.chpl
parth2@ada:~/Desktop/parallelProgramming/lab9$ ./domainmap --numLocales=5
a:  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
b1: 0 0 0 0 1 1 1 2 2 2 3 3 3 4 4 4
b2: 0 0 0 0 1 1 1 2 2 2 3 3 3 4 4 4
b3: 0 0 0 1 1 2 2 2 3 3 4 4 4 4 4 4
parth2@ada:~/Desktop/parallelProgramming/lab9$
```

Question-4: -

The code for mmul2.chpl is: -

```
parth2@ada:~/Desktop/parallelProgramming/lab9$ cat mmul2.chpl
//------------------------------------------------------------------------
// Program code for CS 415/515 Parallel Programming, Portland State University.
//------------------------------------------------------------------------

// Matrix multiplication (shared-memory version)
//   - approach 1: with loops
//   - approach 2: with reduce
//
use BlockDist;

config const N = 8;
var D = {1..N, 1..N};
const D1 = D dmapped Block(D);
const D2 = D dmapped Block({1..N, 1..N});
const D3 = D dmapped Block({1..N, 1..N});

var a: [D] int;
var b: [D1] int;
var c: [D2] int;
var d: [D3] int;

//var a, b, c, d: [D] int;

// initialization
a = [(i,j) in D] i + j;
b = [(i,j) in D] j;
c = 0;
d = 0;

// approach 1
forall (i,j) in D do
  for k in 1..N do
    c(i,j) += a(i,k) * b(k,j);

// approach 2
d = [(i,j) in D] + reduce [k in 1..N] a(i,k) * b(k,j);

writeln("total = ", + reduce c);
writeln("total = ", + reduce d);

//code for verification
write("Locale:");
for (i,j) in c.domain do
writef("%2i", c(i,j).locale.id);
writeln();
parth2@ada:~/Desktop/parallelProgramming/lab9$
```

The output of this code is given below: -

```
OpenSSH SSH client
parth2@ada:~/Desktop/parallelProgramming/lab9$ ./mmul2 --numLocales=5
total = 20736
total = 20736
Locale: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4
parth2@ada:~/Desktop/parallelProgramming/lab9$
```

Changing the number of locales, we can see that the output being generated is correct

```
OpenSSH SSH client
parth2@ada:~/Desktop/parallelProgramming/lab9$ ./mmul2 --numLocales=3
total = 20736
total = 20736
Locale: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
parth2@ada:~/Desktop/parallelProgramming/lab9$
```

Question-5: - When the jacobi.chpl file is run, then

OpenSSH SSH client

```
parth2@ada:~/Desktop/parallelProgramming/lab9$ cat jacobi.chpl
//------------------------------------------------------------------------
// Program code for CS 415/515 Parallel Programming, Portland State University.
//------------------------------------------------------------------------

// Jacobi iteration (shared-memory version)
//

config const N = 8;
const D = {1..N, 1..N};
const PD = D.expand(1,1);       // padded domain
var a:[PD] real, na:[PD] real; // two buffers for a

// formatted 2D array printing
proc print(a:[]) {
  for i in a.domain.dim(0) {
    for j in a.domain.dim(1) do
      writef("%5.2dr", a[i,j]);
    writeln();
  }
}

// initialize boundary conditions
a[PD.dim(0), 0] = 1.0;
a[0, PD.dim(1)] = 2.0;
na[PD.dim(0), 0] = 1.0;
na[0, PD.dim(1)] = 2.0;
writeln("Init:");
print(a);

config const epsilon = 0.01;  // convergence tolerance
config const verbose = false; // printing control

var delta: real;      // for tracking convergence
var t = 0;            // iteration count
do {
  forall ij in D do  // no need to update padded elements
    na[ij] = (a[ij+(0,1)]+a[ij+(0,-1)]+a[ij+(1,0)]+a[ij+(-1,0)])/4.0;
  t += 1;
  delta = max reduce abs(na - a);
  a = na;
  if (verbose) {
    writef("Jacobi iter %i (delta=%5.2r):\n", t, delta);
    print(a);
  }
} while (delta > epsilon);

writeln("Result: t=", t);
print(a);
parth2@ada:~/Desktop/parallelProgramming/lab9$
```

```
parth2@ada:~/Desktop/parallelProgramming/lab9$ vi gausssiedel.chpl
parth2@ada:~/Desktop/parallelProgramming/lab9$ ./jacobi --numLocales=5
Init:
 2.00 2.00 2.00 2.00 2.00 2.00 2.00 2.00 2.00 2.00
 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
Result: t=33
 2.00 2.00 2.00 2.00 2.00 2.00 2.00 2.00 2.00 2.00
 1.00 1.44 1.58 1.61 1.61 1.57 1.49 1.33 0.97 0.00
 1.00 1.19 1.27 1.28 1.26 1.19 1.08 0.88 0.54 0.00
 1.00 1.04 1.05 1.02 0.97 0.88 0.76 0.59 0.33 0.00
 1.00 0.95 0.88 0.81 0.74 0.65 0.54 0.39 0.21 0.00
 1.00 0.87 0.75 0.65 0.56 0.47 0.37 0.26 0.14 0.00
 1.00 0.80 0.63 0.51 0.41 0.33 0.25 0.17 0.09 0.00
 1.00 0.69 0.49 0.36 0.28 0.21 0.15 0.10 0.05 0.00
 1.00 0.50 0.29 0.20 0.14 0.10 0.07 0.05 0.02 0.00
 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
parth2@ada:~/Desktop/parallelProgramming/lab9$
parth2@ada:~/Desktop/parallelProgramming/lab9$
parth2@ada:~/Desktop/parallelProgramming/lab9$
parth2@ada:~/Desktop/parallelProgramming/lab9$
parth2@ada:~/Desktop/parallelProgramming/lab9$
```

Now, for gauss-seidel, we need one array and we will use one buffer and then for tracking convergence, we can use a temporary variable to store the values of a.

The code for the same is given below: -

```
parth2@ada:~/Desktop/parallelProgramming/lab9$ cat gausssiedel.chpl
config const N = 8;
const D = {1..N, 1..N};
const PD = D.expand(1,1);
var a:[PD] real;
var temp:[PD] real;
proc print(a:[]) {
  for i in a.domain.dim(0) {
    for j in a.domain.dim(1) do
      writef("%5.2dr", a[i,j]);
    writeln();
  }
}

a[PD.dim(0), 0] = 1.0;
a[0, PD.dim(1)] = 2.0;
writeln("Init:");
print(a);

config const epsilon = 0.01;
config const verbose = false;

var delta: real;
var t = 0;
temp = a;
do {
  forall ij in D do{   // no need to update padded elements
    temp = a; //storing the older values
    a[ij] = (a[ij+(0,1)]+a[ij+(0,-1)]+a[ij+(1,0)]+a[ij+(-1,0)])/4.0;
  }
  t += 1;
  delta = max reduce abs(a-temp);
  if (verbose) {
    print(a);
  }
} while (delta > epsilon);

writeln("Result: t=", t);
print(a);
parth2@ada:~/Desktop/parallelProgramming/lab9$
```

The output of this program is given below: -

```
parth2@ada:~/Desktop/parallelProgramming/lab9$
parth2@ada:~/Desktop/parallelProgramming/lab9$ ./gausssiedel --numLocales=5
Init:
 2.00 2.00 2.00 2.00 2.00 2.00 2.00 2.00 2.00 2.00
 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
Result: t=12
 2.00 2.00 2.00 2.00 2.00 2.00 2.00 2.00 2.00 2.00
 1.00 1.41 1.54 1.57 1.56 1.53 1.46 1.31 0.96 0.00
 1.00 1.15 1.20 1.21 1.18 1.13 1.03 0.85 0.53 0.00
 1.00 0.98 0.95 0.91 0.86 0.79 0.69 0.54 0.32 0.00
 1.00 0.89 0.80 0.71 0.64 0.56 0.47 0.36 0.20 0.00
 1.00 0.81 0.66 0.54 0.45 0.37 0.30 0.22 0.12 0.00
 1.00 0.76 0.57 0.43 0.33 0.26 0.20 0.14 0.08 0.00
 1.00 0.66 0.44 0.30 0.21 0.16 0.11 0.08 0.04 0.00
 1.00 0.48 0.27 0.17 0.11 0.08 0.06 0.04 0.02 0.00
 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
```

The output was still not converging properly as when I ran it with different numLocales, it showed different outputs.

Then the next iteration of the code which I tried was: -

```
parth2@ada:~/Desktop/parallelProgramming/lab9$ cat gausssiedel.chpl
config const N = 8;
const D = {1..N, 1..N};
const PD = D.expand(1,1);
var a:[PD] real;
proc print(a:[]) {
  for i in a.domain.dim(0) {
    for j in a.domain.dim(1) do
      writef("%5.2dr", a[i,j]);
    writeln();
  }
}

a[PD.dim(0), 0] = 1.0;
a[0, PD.dim(1)] = 2.0;
writeln("Init:");
print(a);

config const epsilon = 0.01;
config const verbose = false;

var delta: real;
var t = 0;
do {
  forall ij in D do{  // no need to update padded elements
    a[ij] = (a[ij+(0,1)]+a[ij+(0,-1)]+a[ij+(1,0)]+a[ij+(-1,0)])/4.0;
  }
  t += 1;
  if (verbose) {
    print(a);
  }
} while (delta > epsilon);

writeln("Result: t=", t);
print(a);
parth2@ada:~/Desktop/parallelProgramming/lab9$
```

The output for this also did not converge properly as given below: -

```
parth2@ada:~/Desktop/parallelProgramming/lab9$ ./gausssiedel --numLocales=5
Init:
 2.00 2.00 2.00 2.00 2.00 2.00 2.00 2.00 2.00 2.00
 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
Result: t=1
 2.00 2.00 2.00 2.00 2.00 2.00 2.00 2.00 2.00 2.00
 1.00 0.75 0.69 0.67 0.67 0.67 0.67 0.67 0.67 0.00
 1.00 0.50 0.31 0.25 0.23 0.22 0.22 0.22 0.22 0.00
 1.00 0.25 0.06 0.02 0.00 0.00 0.00 0.00 0.06 0.00
 1.00 0.38 0.12 0.04 0.01 0.00 0.00 0.00 0.01 0.00
 1.00 0.25 0.06 0.02 0.00 0.00 0.00 0.00 0.00 0.00
 1.00 0.38 0.12 0.04 0.01 0.00 0.00 0.00 0.00 0.00
 1.00 0.25 0.06 0.02 0.00 0.00 0.00 0.00 0.00 0.00
 1.00 0.31 0.09 0.03 0.01 0.00 0.00 0.00 0.00 0.00
 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
parth2@ada:~/Desktop/parallelProgramming/lab9$
parth2@ada:~/Desktop/parallelProgramming/lab9$
parth2@ada:~/Desktop/parallelProgramming/lab9$
parth2@ada:~/Desktop/parallelProgramming/lab9$
```

Question-6: - When FileIO.chpl is run, then we see the following output

```
parth2@ada:~/Desktop/parallelProgramming/lab9$ chpl -g -o fileIO fileIO.chpl
parth2@ada:~/Desktop/parallelProgramming/lab9$ ./fileIO --numLocales=5
N=16, a: 731 7696 6176 2657 206 3218 4698 4015 6518 7697 2098 7560 1141 6176 2171 3541
N=16, b: 731 7696 6176 2657 206 3218 4698 4015 6518 7697 2098 7560 1141 6176 2171 3541
parth2@ada:~/Desktop/parallelProgramming/lab9$ cat
```

The program that produced this output is: -

```
parth2@ada:~/Desktop/parallelProgramming/lab9$ cat fileIO.chpl
//-----------------------------------------------------------------------------
// Program code for CS 415/515 Parallel Programming, Portland State University.
//-----------------------------------------------------------------------------

// File I/O examples
//
use IO;

config const infile = "input";
config const outfile = "output";
const fin = open(infile, iomode.r);
const fout = open(outfile, iomode.cw);

const N = fin.size/4;        // find integer count of infile
const D = {1..N};
var a:[D] int[32];           // note the size specification

// file input
var r = fin.reader(kind=ionative);  // create a channel for binary code
r.read(a);                   // read into array a
r.close();
writeln("N=", N, ", a: ", a);

// file output
var w = fout.writer(kind=ionative);
w.write(a);
w.close();

// parallel input
use BlockDist;
const MD = D dmapped Block(D);
var b:[MD] int[32];
forall i in MD do {
  var r = fin.reader(kind=ionative, start=i*4-4, end=i*4);
  r.read(b[i]);
  r.close();
}
writeln("N=", N, ", b: ", b);

fin.close();
fout.close();


parth2@ada:~/Desktop/parallelProgramming/lab9$
```

The changes needed to be made for the required for N=32 are:-

Change the int type as well as the calculation of N by removing the divisor by half.

The code becomes as: -

OpenSSH SSH client

```
parth2@ada:~/Desktop/parallelProgramming/lab9$ cat fileIO.chpl
//------------------------------------------------------------------------
// Program code for CS 415/515 Parallel Programming, Portland State University.
//------------------------------------------------------------------------

// File I/O examples
//
use IO;

config const infile = "input";
config const outfile = "output";
const fin = open(infile, iomode.r);
const fout = open(outfile, iomode.cw);

const N = fin.size/2;       // find integer count of infile
const D = {1..N};
var a:[D] uint[8];          // note the size specification

// file input
var r = fin.reader(kind=ionative);  // create a channel for binary code
r.read(a);                  // read into array a
r.close();
writeln("N=", N, ", a: ", a);

// file output
var w = fout.writer(kind=ionative);
w.write(a);
w.close();

// parallel input
use BlockDist;
const MD = D dmapped Block(D);
var b:[MD] int[32];
forall i in MD do {
  var r = fin.reader(kind=ionative, start=i*4-4, end=i*4);
  r.read(b[i]);
  r.close();
}
writeln("N=", N, ", b: ", b);

fin.close();
fout.close();


parth2@ada:~/Desktop/parallelProgramming/lab9$
```

The output becomes as: -

OpenSSH SSH client

```
parth2@ada:~/Desktop/parallelProgramming/lab9$ vi fileIO.chpl
parth2@ada:~/Desktop/parallelProgramming/lab9$ chpl -g -o fileIO fileIO.chpl
parth2@ada:~/Desktop/parallelProgramming/lab9$ ./fileIO --numLocales=2
N=32, a: 219 2 0 0 16 30 0 0 32 24 0 0 97 10 0 0 206 0 0 0 146 12 0 0 90 18 0 0 175 15 0 0
N=32, b: 731 7696 6176 2657 206 3218 4698 4015 6518 7697 2098 7560 1141 6176 2171 3541 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
parth2@ada:~/Desktop/parallelProgramming/lab9$
```

Now for achieving the required 64 integers, we will modify the calculation of N as follows: -

```
parth2@ada:~/Desktop/parallelProgramming/lab9$
parth2@ada:~/Desktop/parallelProgramming/lab9$ cat fileIO.chpl
//-------------------------------------------------------------------------
// Program code for CS 415/515 Parallel Programming, Portland State University.
//-------------------------------------------------------------------------

// File I/O examples
//
use IO;

config const infile = "input";
config const outfile = "output";
const fin = open(infile, iomode.r);
const fout = open(outfile, iomode.cw);

const N = fin.size;       // find integer count of infile
const D = {1..N};
var a:[D] uint[8];          // note the size specification

// file input
var r = fin.reader(kind=ionative);  // create a channel for binary code
r.read(a);                    // read into array a
r.close();
writeln("N=", N, ", a: ", a);

// file output
var w = fout.writer(kind=ionative);
w.write(a);
w.close();

// parallel input
use BlockDist;
const MD = D dmapped Block(D);
var b:[MD] int[32];
forall i in MD do {
  var r = fin.reader(kind=ionative, start=i*4-4, end=i*4);
  r.read(b[i]);
  r.close();
}
writeln("N=", N, ", b: ", b);

fin.close();
fout.close();
```

The output for this code is given below: -

```
OpenSSH SSH client
parth2@ada:~/Desktop/parallelProgramming/lab9$ chpl -g -o fileIO fileIO.chpl
parth2@ada:~/Desktop/parallelProgramming/lab9$ ./fileIO --numLocales=2
N=64, a: 219 2 0 0 16 30 0 0 32 24 0 0 97 10 0 0 206 0 0 0 146 12 0 0 90 18 0 0 175 15 0 0 118 25 0 0 17 30 0 0 50 8 0 0 136 29 0 0 117 4 0 0 32 24 0 0 123 8 0 0 213 13 0 0
N=64, b: 731 7696 6176 2657 206 3218 4698 4015 6518 7697 2098 7560 1141 6176 2171 3541 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
parth2@ada:~/Desktop/parallelProgramming/lab9$
parth2@ada:~/Desktop/parallelProgramming/lab9$
parth2@ada:~/Desktop/parallelProgramming/lab9$
parth2@ada:~/Desktop/parallelProgramming/lab9$ cat fileIO.chpl
//-------------------------------------------------------------------------
```

If we want the type of integers to be normal 32 bit integers, then the code can be modified as below: -

OpenSSH SSH client

```
//-----------------------------------------------------------------------------
// Program code for CS 415/515 Parallel Programming, Portland State University.
//-----------------------------------------------------------------------------

// File I/O examples
//
use IO;

config const infile = "input";
config const outfile = "output";
const fin = open(infile, iomode.r);
const fout = open(outfile, iomode.cw);

const N = fin.size;       // find integer count of infile
const D = {1..N};
var a:[D] uint[32];          // note the size specification

// file input
var r = fin.reader(kind=ionative);  // create a channel for binary code
r.read(a);                    // read into array a
r.close();
writeln("N=", N, ", a: ", a);

// file output
var w = fout.writer(kind=ionative);
w.write(a);
w.close();

// parallel input
use BlockDist;
const MD = D dmapped Block(D);
var b:[MD] int[32];
forall i in MD do {
  var r = fin.reader(kind=ionative, start=i*4-4, end=i*4);
  r.read(b[i]);
  r.close();
}
writeln("N=", N, ", b: ", b);

fin.close();
fout.close();
```

The output for this code is given below: -

```
OpenSSH SSH client
parth2@ada:~/Desktop/parallelProgramming/lab9$ vi fileIO.chpl
parth2@ada:~/Desktop/parallelProgramming/lab9$ chpl -g -o fileIO fileIO.chpl
parth2@ada:~/Desktop/parallelProgramming/lab9$ ./fileIO --numLocales=2
N=64, a: 731 7696 6176 2657 206 3218 4698 4015 6518 7697 2098 7560 1141 6176 2171 3541 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
N=64, b: 731 7696 6176 2657 206 3218 4698 4015 6518 7697 2098 7560 1141 6176 2171 3541 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
parth2@ada:~/Desktop/parallelProgramming/lab9$
parth2@ada:~/Desktop/parallelProgramming/lab9$
parth2@ada:~/Desktop/parallelProgramming/lab9$
parth2@ada:~/Desktop/parallelProgramming/lab9$
```

Question-7: - fileIO2.chpl

The code for the same is given below: -

OpenSSH SSH client

```
parth2@ada:~/Desktop/parallelProgramming/lab9$ cat fileIO2.chpl
//------------------------------------------------------------------------
// Program code for CS 415/515 Parallel Programming, Portland State University.
//------------------------------------------------------------------------

// File I/O examples
//
use IO;

config const infile = "input";
config const outfile = "output";
const fin = open(infile, iomode.r);
const fout = open(outfile, iomode.cw);

const N = fin.size;       // find integer count of infile
const D = {1..N};
var a:[D] uint[32];          // note the size specification

// file input
var r = fin.reader(kind=ionative);  // create a channel for binary code
r.read(a);                    // read into array a
r.close();
writeln("N=", N, ", a: ", a);

// file output
var w = fout.writer(kind=ionative);
w.write(a);
w.close();

// parallel input
use BlockDist;
const MD = D dmapped Block(D);
var b:[MD] int[32];
forall i in MD do {
  worker();
  var r = fin.reader(kind=ionative, start=i*4-4, end=i*4);
  r.read(b[i]);
  r.close();
}
writeln("N=", N, ", b: ", b);

fin.close();
fout.close();

for loc in Locales do
on loc do worker();
parth2@ada:~/Desktop/parallelProgramming/lab9$
```