Exercise 4

Serial code:

```
/* This code is contributed by Richard T. Evans at the Texas Advanced computing Center
 * The University of Texas at Austin
 *
 * To compile: icc -o heat heat_serial.c calc_up.c
*/

#include <stdio.h>
#include <sys/time.h>
#include "calc_up.h"

int main() {

  int Nx,Ny,Nt;
  int t,x,y;
  Nx=1000;
  Ny=1000;
  Nt=1000;
  double u[Nx][Ny];
  double up[Nx][Ny];

  struct timeval start,end;
  float delta;

  // Boundary conditions
  for(x=0; x<Nx; x++)
    for(y=0; y<Ny; y++) {
      if (x==0)
     u[x][y] = 1.0;
      else
     u[x][y] = 0.0;
    }

  gettimeofday(&start, NULL);



////////////////////////////////////////////////////////////////////
// Finite difference algorithm - iterate over time to reach steady
state
```

```
//////////////////////////////////////////////////////////////////////
//
  for(t=0;t<Nt;t++) {
    for(x=1; x<Nx-1; x++)
      for(y=1; y<Ny-1; y++)
     calc_up(x,y,Nx,Ny,u,up);

    for(x=1; x<Nx-1; x++)
      for(y=1; y<Ny-1; y++)
     u[x][y] = up[x][y];
  }

  gettimeofday(&end, NULL);
  delta = ((end.tv_sec-start.tv_sec)*1000000u + end.tv_usec-
start.tv_usec)/1.e6;


  double sum = 0;
  for(y=0; y<Ny; y++) {
    for(x=0; x<Nx; x++) {
      sum += u[x][y];
    }
  }


  printf("run time    = %fs\n", delta);
  printf("sum of u    = %f\n",  sum);
}
```

Ritu Arora 9/13/2017 9:46 AM
Comment [1]: Parallelize this loop

Ritu Arora 9/13/2017 9:46 AM
Comment [2]: Parallelize this loop

Steps for Using IPT

c557-903$ ../../IPT heat_serial.c

NOTE: We currently support only C and C++ programs.

Please select a parallel programming model from the following available options:
1. MPI
2. OpenMP
3. CUDA
2

NOTE: As per the OpenMP standard, a parallelized region/block of statements can have only one entry point and only one exit point. Branching out or breaking prematurely from a parallelized region/block of statements is not allowed. Please make sure that there are no return/break statements in the region selected for parallelization. However, exit/continue statements are allowed in parallel regions.

A list containing the functions in the input file will be presented, and you may want to select one function at a time to parallelize it using multi-threading.

Please choose the function that you want to parallelize from the list below
1 : main
1

Please select one of the following options (enter 1 or 2 or 3)
1. Create a parallel region (a group of threads will be created and each thread will execute a block of code redundantly but in parallel)
2. Parallelize a for-loop (a group of threads will be created and each thread will execute a certain number of iterations of a for-loop)
3. Create a parallel section (TBD - this mode is currently unavailable)
2

Note: With your response, you will be selecting or declining the parallelization of the outermost for-loop in the code region shown below. If instead of the outermost for-loop, there are any inner for-loops in this code region that you are interested in parallelizing, then, you will be able to select those at a later stage.

```
// Boundary conditions
for (x = 0; x < Nx; x++)
  for (y = 0; y < Ny; y++) {
    if (x == 0)
      u[x][y] = 1.0;
    else
      u[x][y] = 0.0;
  }
```
Is this the for loop you are looking for?(y/n)
n

OK - will find the next loop if available.

```
for (y = 0; y < Ny; y++) {
  if (x == 0)
    u[x][y] = 1.0;
```

```
  else
    u[x][y] = 0.0;
}
```
Is this the for loop you are looking for?(y/n)
n

OK - will find the next loop if available.

Note: With your response, you will be selecting or declining the parallelization of the outermost for-loop in the code region shown below. If instead of the outermost for-loop, there are any inner for-loops in this code region that you are interested in parallelizing, then, you will be able to select those at a later stage.

```
//////////////////////////////////////////////////////////////////////
// Finite difference algorithm - iterate over time to reach steady state
//////////////////////////////////////////////////////////////////////
for (t = 0; t < Nt; t++) {
  for (x = 1; x < (Nx - 1); x++)
    for (y = 1; y < (Ny - 1); y++)
      calc_up(x,y,Nx,Ny,u,up);
  for (x = 1; x < (Nx - 1); x++)
    for (y = 1; y < (Ny - 1); y++)
      u[x][y] = up[x][y];
}
```
Is this the for loop you are looking for?(y/n)
n

OK - will find the next loop if available.

Note: With your response, you will be selecting or declining the parallelization of the outermost for-loop in the code region shown below. If instead of the outermost for-loop, there are any inner for-loops in this code region that you are interested in parallelizing, then, you will be able to select those at a later stage.

```
for (x = 1; x < (Nx - 1); x++)
  for (y = 1; y < (Ny - 1); y++)
    calc_up(x,y,Nx,Ny,u,up);
```
Is this the for loop you are looking for?(y/n)
y

Reduction variables are the variables that should be updated by the OpenMP threads and then accumulated according to a mathematical operation like sum, multiplication,etc.

Do you want to perform reduction on any variable ?(Y/N)
n

Are there any lines of code that you would like to run either using a single thread at a time (hence, one thread after another), or using only one thread?(Y/N)
n

Would you like to parallelize another loop in the previously selected function or another one?(Y/N)
y

Please choose the function that you want to parallelize from the list below
1 : main
1

Note: With your response, you will be selecting or declining the parallelization of the outermost for-loop in the code region shown below. If instead of the outermost for-loop, there are any inner for-loops in this code region that you are interested in parallelizing, then, you will be able to select those at a later stage.

```
// Boundary conditions
for (x = 0; x < Nx; x++)
  for (y = 0; y < Ny; y++) {
    if (x == 0)
      u[x][y] = 1.0;
    else
      u[x][y] = 0.0;
  }
```

Is this the for loop you are looking for?(y/n)
n

OK - will find the next loop if available.

```
for (y = 0; y < Ny; y++) {
  if (x == 0)
    u[x][y] = 1.0;
  else
    u[x][y] = 0.0;
}
```

Is this the for loop you are looking for?(y/n)
n

OK - will find the next loop if available.

Note: With your response, you will be selecting or declining the parallelization of the outermost for-loop in the code region shown below. If instead of the outermost for-loop, there are any inner for-loops in this code region that you are interested in parallelizing, then, you will be able to select those at a later stage.

```
////////////////////////////////////////////////////////////////////////
// Finite difference algorithm - iterate over time to reach steady state
////////////////////////////////////////////////////////////////////////
for (t = 0; t < Nt; t++) {

#pragma omp parallel default(none) shared(u,up,Nx,Ny) private(x,y)
{

#pragma omp for
  for (x = 1; x < (Nx - 1); x++)
    for (y = 1; y < (Ny - 1); y++)
      calc_up(x,y,Nx,Ny,u,up);
}
  for (x = 1; x < (Nx - 1); x++)
    for (y = 1; y < (Ny - 1); y++)
      u[x][y] = up[x][y];
}
```

Is this the for loop you are looking for?(y/n)

n

OK - will find the next loop if available.

```
for (y = 1; y < (Ny - 1); y++)
  calc_up(x,y,Nx,Ny,u,up);
```

Is this the for loop you are looking for?(y/n)

n

OK - will find the next loop if available.

Note: With your response, you will be selecting or declining the parallelization of the outermost for-loop in the code region shown below. If instead of the outermost for-loop, there are any inner for-loops in this code region that you are interested in parallelizing, then, you will be able to select those at a later stage.

```
for (x = 1; x < (Nx - 1); x++)
```

```
  for (y = 1; y < (Ny - 1); y++)
    u[x][y] = up[x][y];
```

Is this the for loop you are looking for?(y/n)
y

Reduction variables are the variables that should be updated by the OpenMP threads and then accumulated according to a mathematical operation like sum, multiplication,etc.

Do you want to perform reduction on any variable ?(Y/N)
n

IPT is unable to perform the dependency analysis of the array named [u] in the region of code that you wish to parallelize. Please enter 1 if the entire array is being updated in a single iteration of the loop that you selected for parallelization, or, enter 2 otherwise.
2

Are there any lines of code that you would like to run either using a single thread at a time (hence, one thread after another), or using only one thread?(Y/N)
n

Would you like to parallelize another loop in the previously selected function or another one?(Y/N)
n

Are you writing/printing anything from the parallelized region of the code?(Y/N)
n

Running Consistency Tests

Compiling and running the generated code

```
c557-903$ ls -ltr
total 44
-rw-r--r-- 1 rauta G-25072  1358 Sep 11 17:01 README.txt
-rw-r--r-- 1 rauta G-25072 14726 Sep 11 17:01 md.c
-rw-r--r-- 1 rauta G-25072  1286 Sep 11 17:01 heat_serial.c
-rw-r--r-- 1 rauta G-25072  5555 Sep 11 17:01 circuit.c
-rw-r--r-- 1 rauta G-25072    81 Sep 11 17:01 calc_up.h
-rw-r--r-- 1 rauta G-25072   184 Sep 11 17:01 calc_up.c
-rw-r--r-- 1 rauta G-25072  1637 Sep 12 21:05 rose_heat_serial_OpenMP.c

c557-903$ icc -qopenmp -o rose_heat_serial_OpenMP rose_heat_serial_OpenMP.c calc_up.c
```

```
c557-903$ ls -ltr
total 45
-rw-r--r-- 1 rauta G-25072  1358 Sep 11 17:01 README.txt
-rw-r--r-- 1 rauta G-25072 14726 Sep 11 17:01 md.c
-rw-r--r-- 1 rauta G-25072  1286 Sep 11 17:01 heat_serial.c
-rw-r--r-- 1 rauta G-25072  5555 Sep 11 17:01 circuit.c
-rw-r--r-- 1 rauta G-25072    81 Sep 11 17:01 calc_up.h
-rw-r--r-- 1 rauta G-25072   184 Sep 11 17:01 calc_up.c
-rw-r--r-- 1 rauta G-25072  1637 Sep 12 21:05 rose_heat_serial_OpenMP.c
-rwxr-xr-x 1 rauta G-25072 62658 Sep 12 21:08 rose_heat_serial_OpenMP

c557-903$ export OMP_NUM_THREADS=1

c557-903$ time ./rose_heat_serial_OpenMP
run time    = 5.042156s
sum of u    = 4075.324785

real    0m5.092s
user    0m5.037s
sys     0m0.008s

c557-903$ export OMP_NUM_THREADS=16

c557-903$ time ./rose_heat_serial_OpenMP
run time    = 0.871802s
sum of u    = 4075.324785

real    0m0.881s
user    0m11.996s
sys     0m0.148s
```