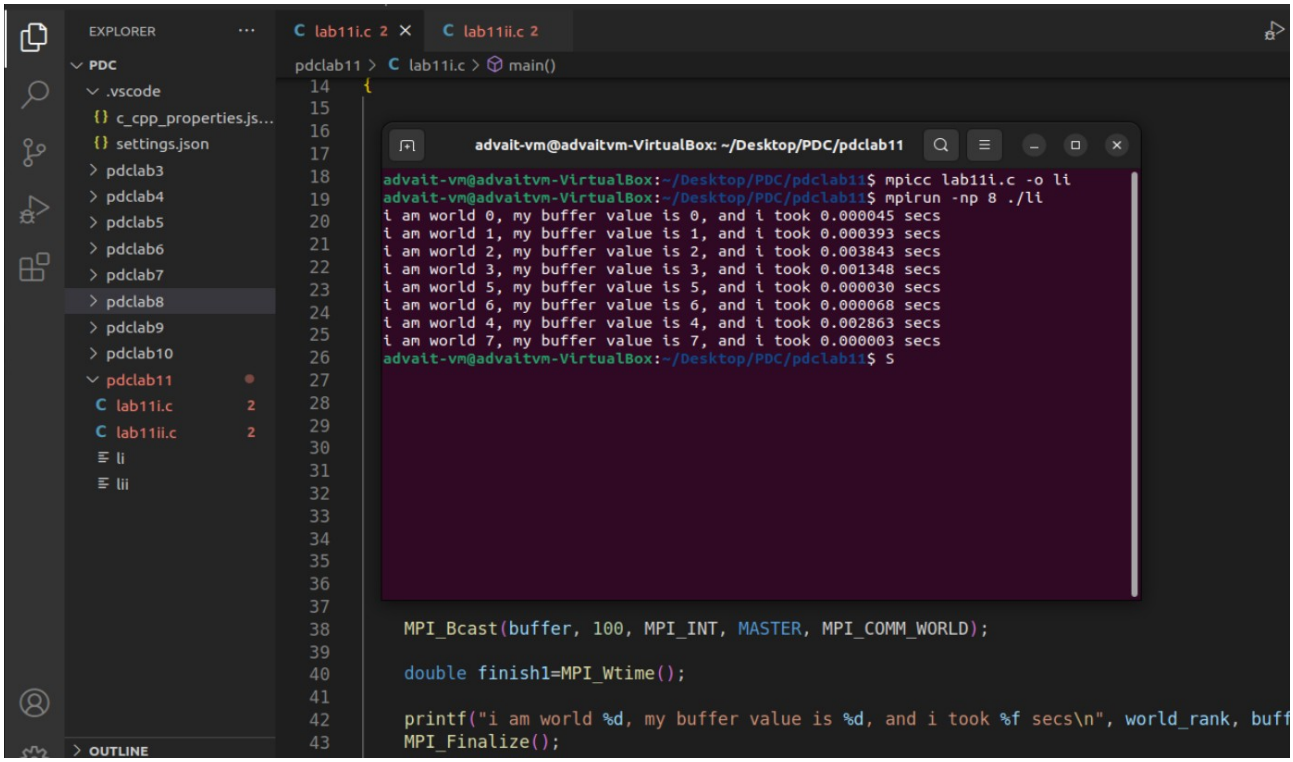
PDC LAB 11

Advait Deochakke

20BCE1143

1. Broadcast



Code:

```
#include<mpi.h>
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<stdbool.h>
#include<unistd.h>

#define MASTER 0
#define FROM_MASTER 1
#define FROM_WORKER 2

int main()
{
```

```
  MPI_Status status;
  MPI_Request request = MPI_REQUEST_NULL;

  MPI_Init(NULL, NULL);

  int world_size;
  MPI_Comm_size(MPI_COMM_WORLD, &world_size);

  int world_rank;
  MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);

  double start = MPI_Wtime();

  int msg_tag=1729;
  int buffer[100]={0};
  if(world_rank==MASTER)
  {
    for(int i=0; i<100; i++)
      buffer[i]=i;
  }
  double start1=MPI_Wtime();

  MPI_Bcast(buffer, 100, MPI_INT, MASTER, MPI_COMM_WORLD);

  double finish1=MPI_Wtime();

  printf("i am world %d, my buffer value is %d, and i took %f secs\n",
world_rank, buffer[world_rank], finish1-start1);
  MPI_Finalize();

  return 0;

}
```

<next Page>

2. Scatter – Example

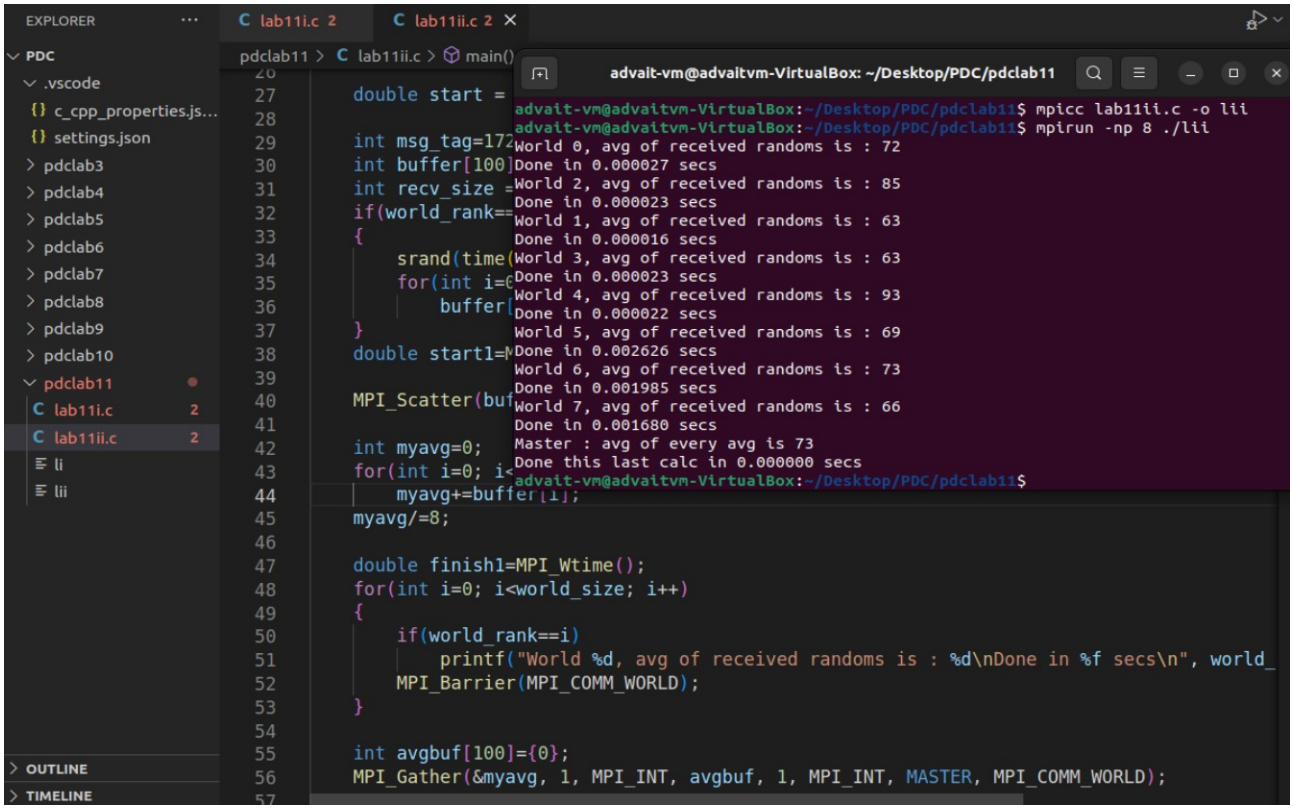
3. Scatter – Random Numbers

4. Gather

5. Barrier

(Combined into One code to show functionality of each)

(Barrier makes sure that printf order is consistent 01234567)



Code :

```
#include<mpi.h>
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<stdbool.h>
#include<unistd.h>

#define MASTER 0
#define FROM_MASTER 1
```

```
#define FROM_WORKER 2

int main()
{

  MPI_Status status;
  MPI_Request request = MPI_REQUEST_NULL;

  MPI_Init(NULL, NULL);

  int world_size;
  MPI_Comm_size(MPI_COMM_WORLD, &world_size);

  int world_rank;
  MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);

  double start = MPI_Wtime();

  int msg_tag=1729;
  int buffer[100]={0};
  int recv_size = 100/world_size;
  if(world_rank==MASTER)
  {
    srand(time(NULL));
    for(int i=0; i<world_size*recv_size; i++)
      buffer[i]=rand()%100;
  }
  double start1=MPI_Wtime();

  MPI_Scatter(buffer, recv_size, MPI_INT, buffer, recv_size, MPI_INT,
MASTER, MPI_COMM_WORLD);

  int myavg=0;
  for(int i=0; i<recv_size; i++)
    myavg+=buffer[i];
  myavg/=8;

  double finish1=MPI_Wtime();
  for(int i=0; i<world_size; i++)
  {
    if(world_rank==i)
      printf("World %d, avg of received randoms is : %d\nDone in %f secs\
n", world_rank, myavg, finish1-start1);
    MPI_Barrier(MPI_COMM_WORLD);
  }

  int avgbuf[100]={0};
  MPI_Gather(&myavg, 1, MPI_INT, avgbuf, 1, MPI_INT, MASTER,
MPI_COMM_WORLD);

  if(world_rank==MASTER)
```

```
  {
    double start=MPI_Wtime();
    int fullavg=0;
    for(int i=0; i<world_size; i++)
      fullavg+=avgbuf[i];
    fullavg/=world_size;
    double finish=MPI_Wtime();
    printf("Master : avg of every avg is %d\nDone this last calc in %f
secs\n", fullavg, finish -start);
  }

  MPI_Finalize();

  return 0;

}
```
