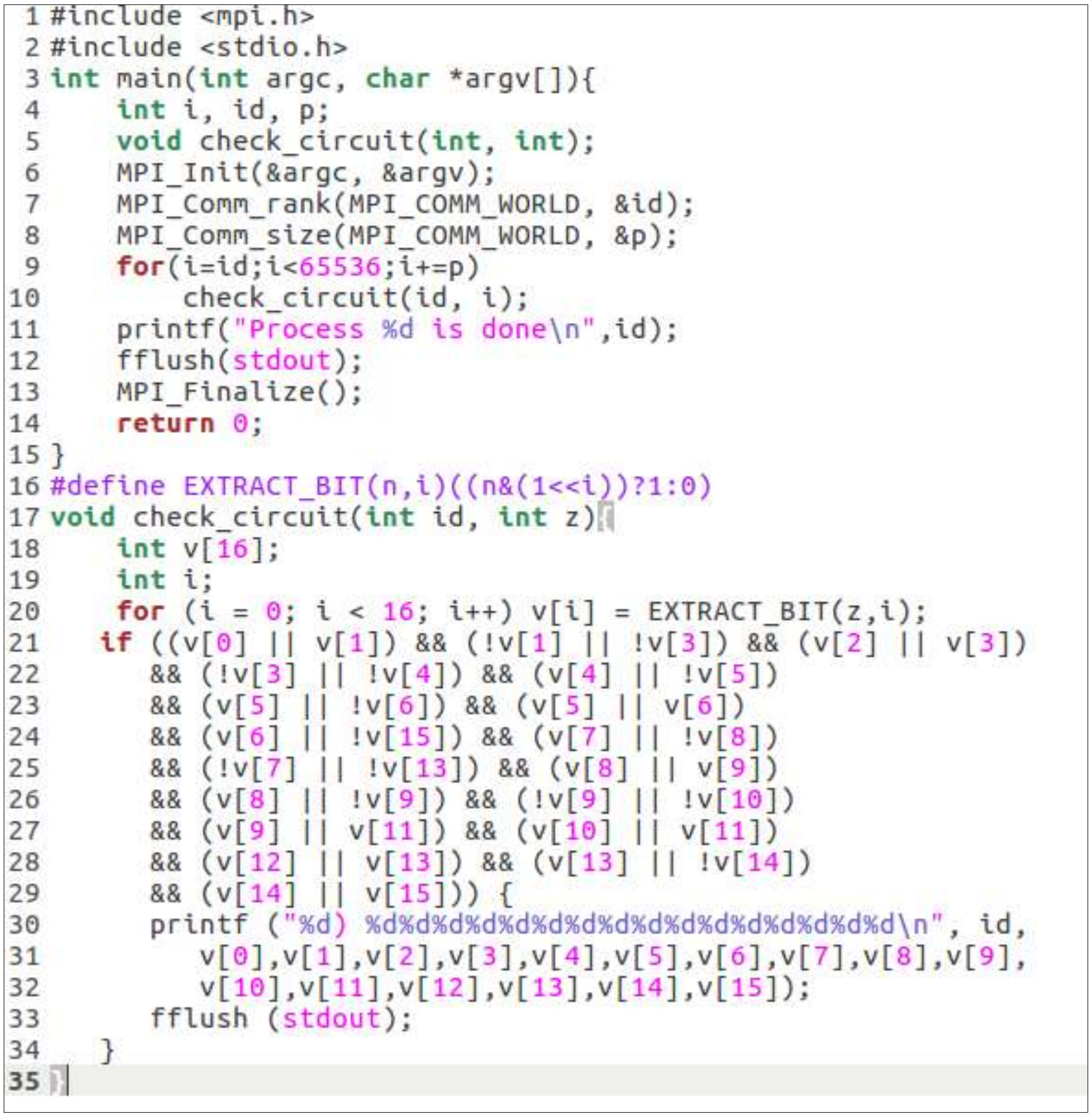
DHRUV GARG 16 BCE 1190

PDC LAB 5 – PROBLEM STATEMENT 2

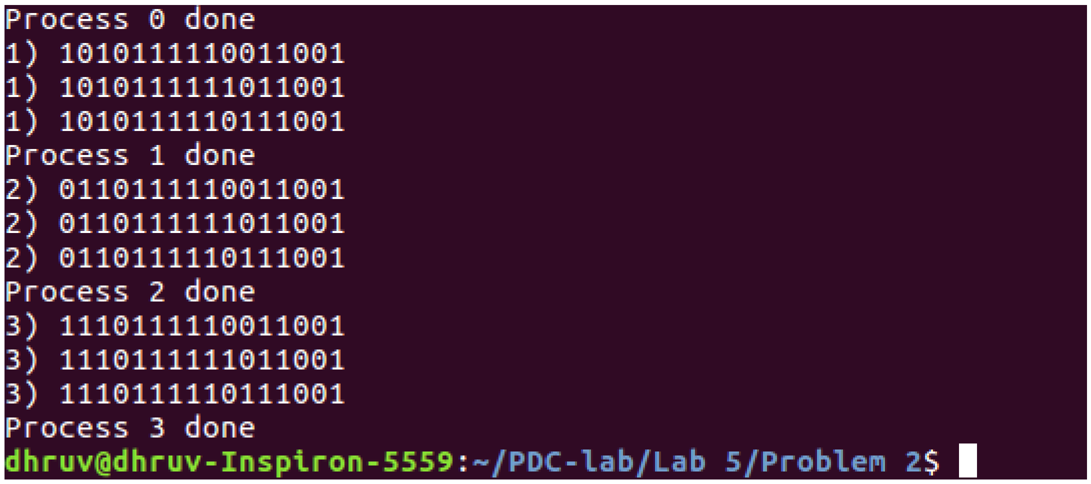
Circuit Satisfiability Problem

Code



```
#include <mpi.h>
#include <stdio.h>
int main(int argc, char *argv[]){
    int i, id, p;
    void check_circuit(int, int);
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &id);
    MPI_Comm_size(MPI_COMM_WORLD, &p);
    for(i=id;i<65536;i+=p)
        check_circuit(id, i);
    printf("Process %d is done\n",id);
    fflush(stdout);
    MPI_Finalize();
    return 0;
}
#define EXTRACT_BIT(n,i)((n&(1<<i))?1:0)
void check_circuit(int id, int z){
    int v[16];
    int i;
    for (i = 0; i < 16; i++) v[i] = EXTRACT_BIT(z,i);
   if ((v[0] || v[1]) && (!v[1] || !v[3]) && (v[2] || v[3])
      && (!v[3] || !v[4]) && (v[4] || !v[5])
      && (v[5] || !v[6]) && (v[5] || v[6])
      && (v[6] || !v[15]) && (v[7] || !v[8])
      && (!v[7] || !v[13]) && (v[8] || v[9])
      && (v[8] || !v[9]) && (!v[9] || !v[10])
      && (v[9] || v[11]) && (v[10] || v[11])
      && (v[12] || v[13]) && (v[13] || !v[14])
      && (v[14] || v[15])) {
      printf ("%d) %d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d\n", id,
         v[0],v[1],v[2],v[3],v[4],v[5],v[6],v[7],v[8],v[9],
         v[10],v[11],v[12],v[13],v[14],v[15]);
      fflush (stdout);
   }
}
```

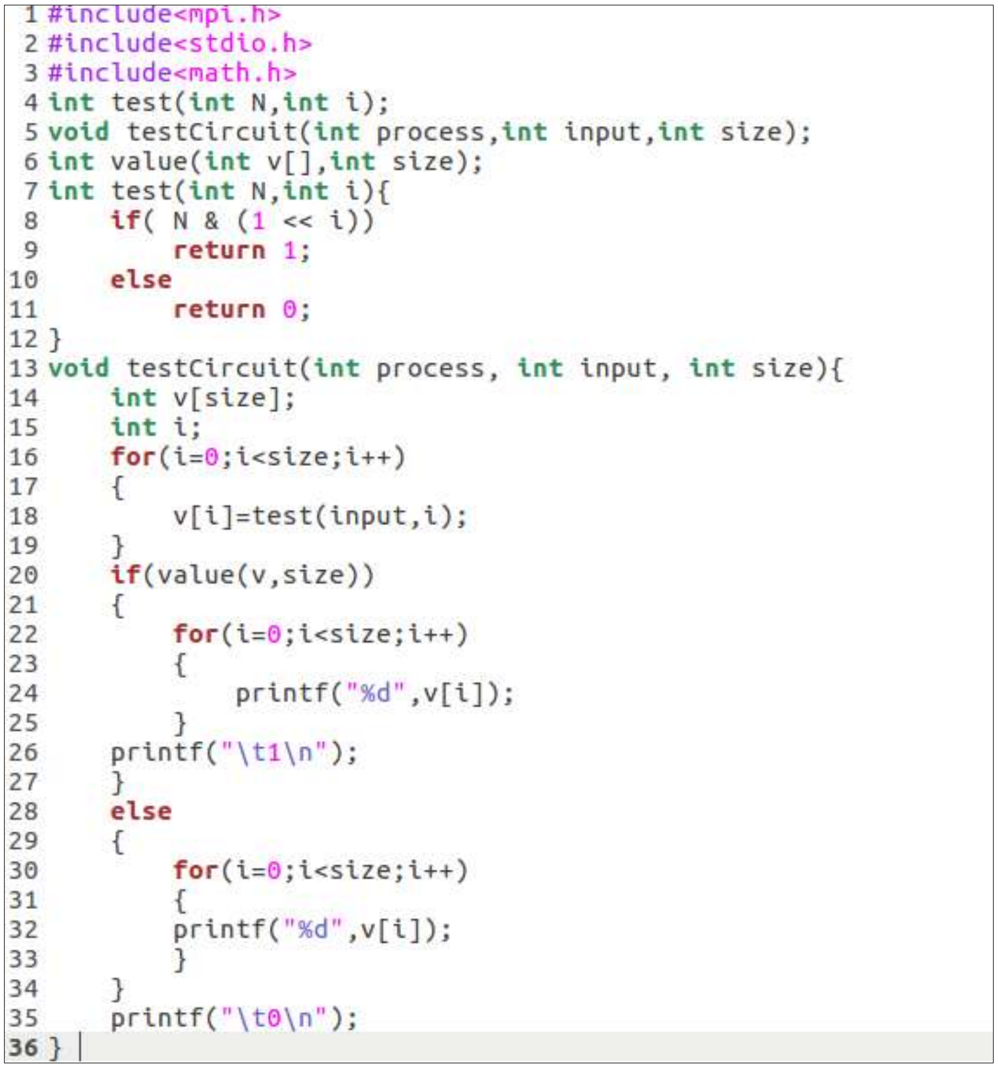
Output



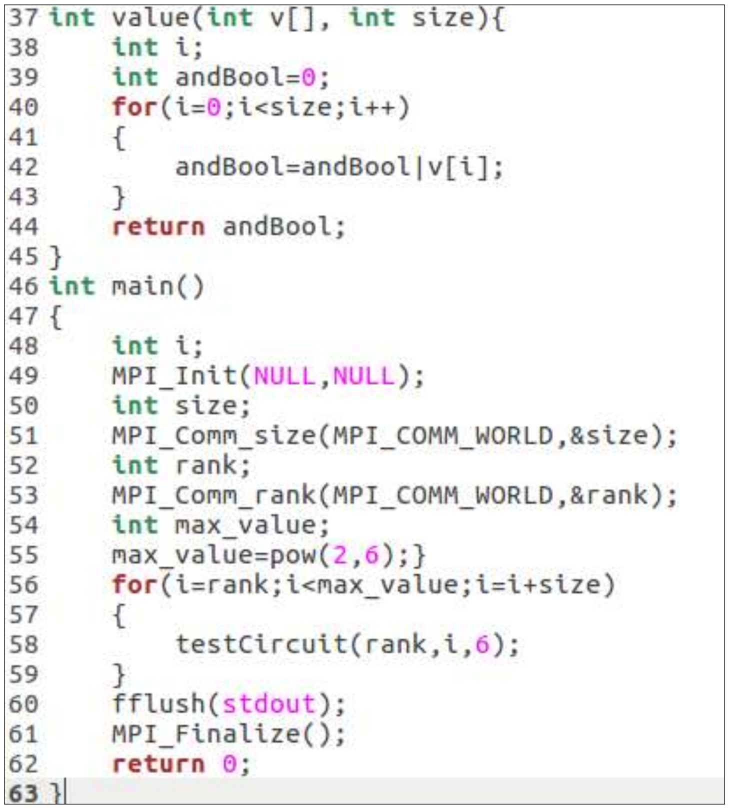
```
Process 0 done
1) 1010111110011001
1) 1010111111011001
1) 1010111110111001
Process 1 done
2) 0110111110011001
2) 0110111111011001
2) 0110111110111001
Process 2 done
3) 1110111110011001
3) 1110111111011001
3) 1110111110111001
Process 3 done
dhruv@dhruv-Inspiron-5559:~/PDC-lab/Lab 5/Problem 2$
```

Truth Table Generation

Code

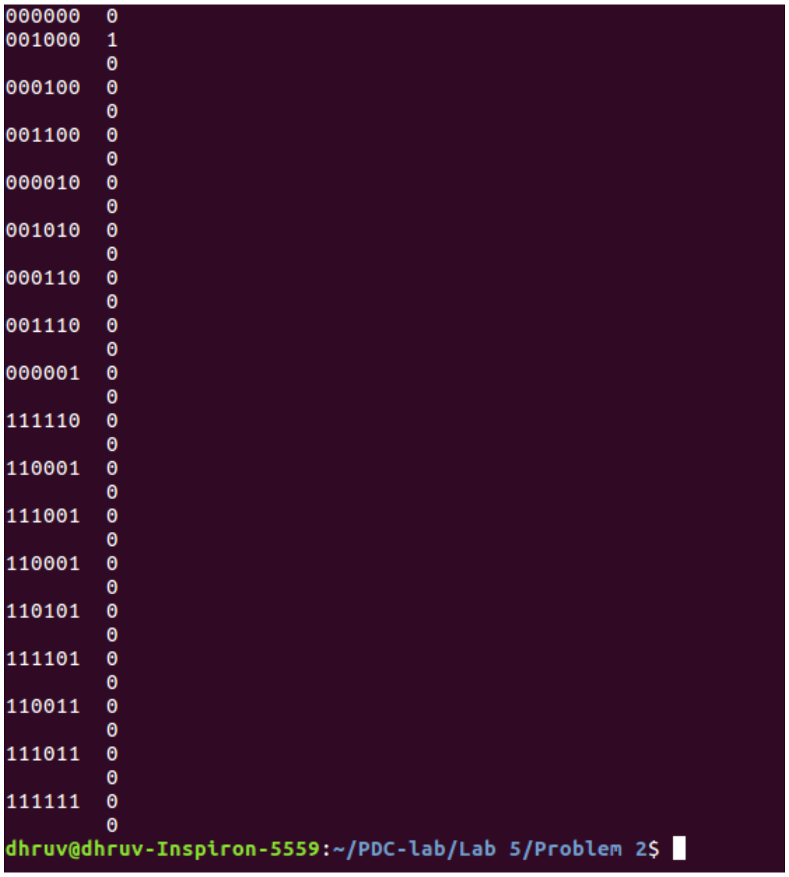


```c
#include<mpi.h>
#include<stdio.h>
#include<math.h>
int test(int N,int i);
void testCircuit(int process,int input,int size);
int value(int v[],int size);
int test(int N,int i){
    if( N & (1 << i))
        return 1;
    else
        return 0;
}
void testCircuit(int process, int input, int size){
    int v[size];
    int i;
    for(i=0;i<size;i++)
    {
        v[i]=test(input,i);
    }
    if(value(v,size))
    {
        for(i=0;i<size;i++)
        {
            printf("%d",v[i]);
        }
    printf("\t1\n");
    }
    else
    {
        for(i=0;i<size;i++)
        {
        printf("%d",v[i]);
        }
    }
    printf("\t0\n");
}
```



```
int value(int v[], int size){
    int i;
    int andBool=0;
    for(i=0;i<size;i++)
    {
        andBool=andBool|v[i];
    }
    return andBool;
}
int main()
{
    int i;
    MPI_Init(NULL,NULL);
    int size;
    MPI_Comm_size(MPI_COMM_WORLD,&size);
    int rank;
    MPI_Comm_rank(MPI_COMM_WORLD,&rank);
    int max_value;
    max_value=pow(2,6);;}
    for(i=rank;i<max_value;i=i+size)
    {
        testCircuit(rank,i,6);
    }
    fflush(stdout);
    MPI_Finalize();
    return 0;
}
```

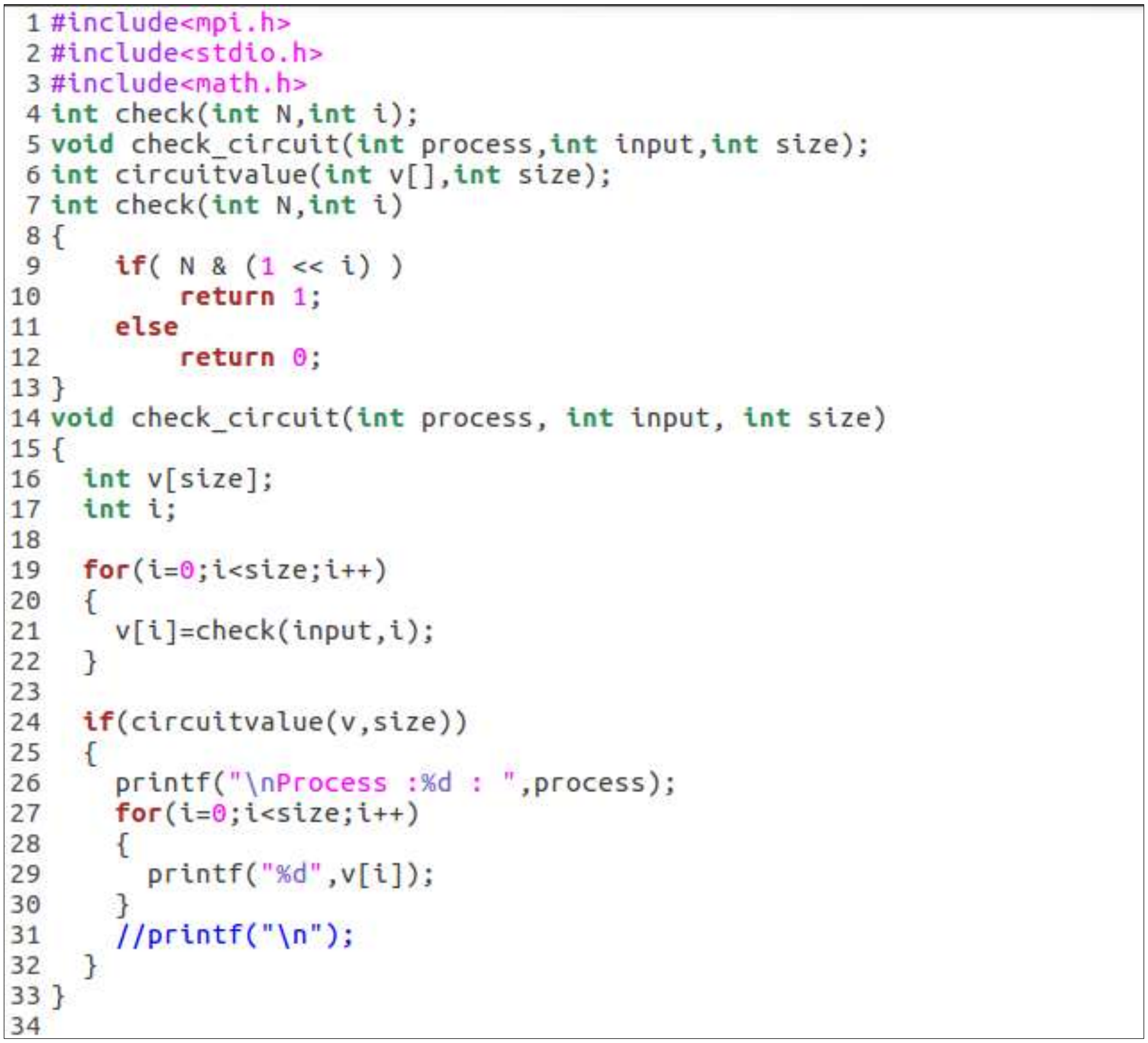
Output



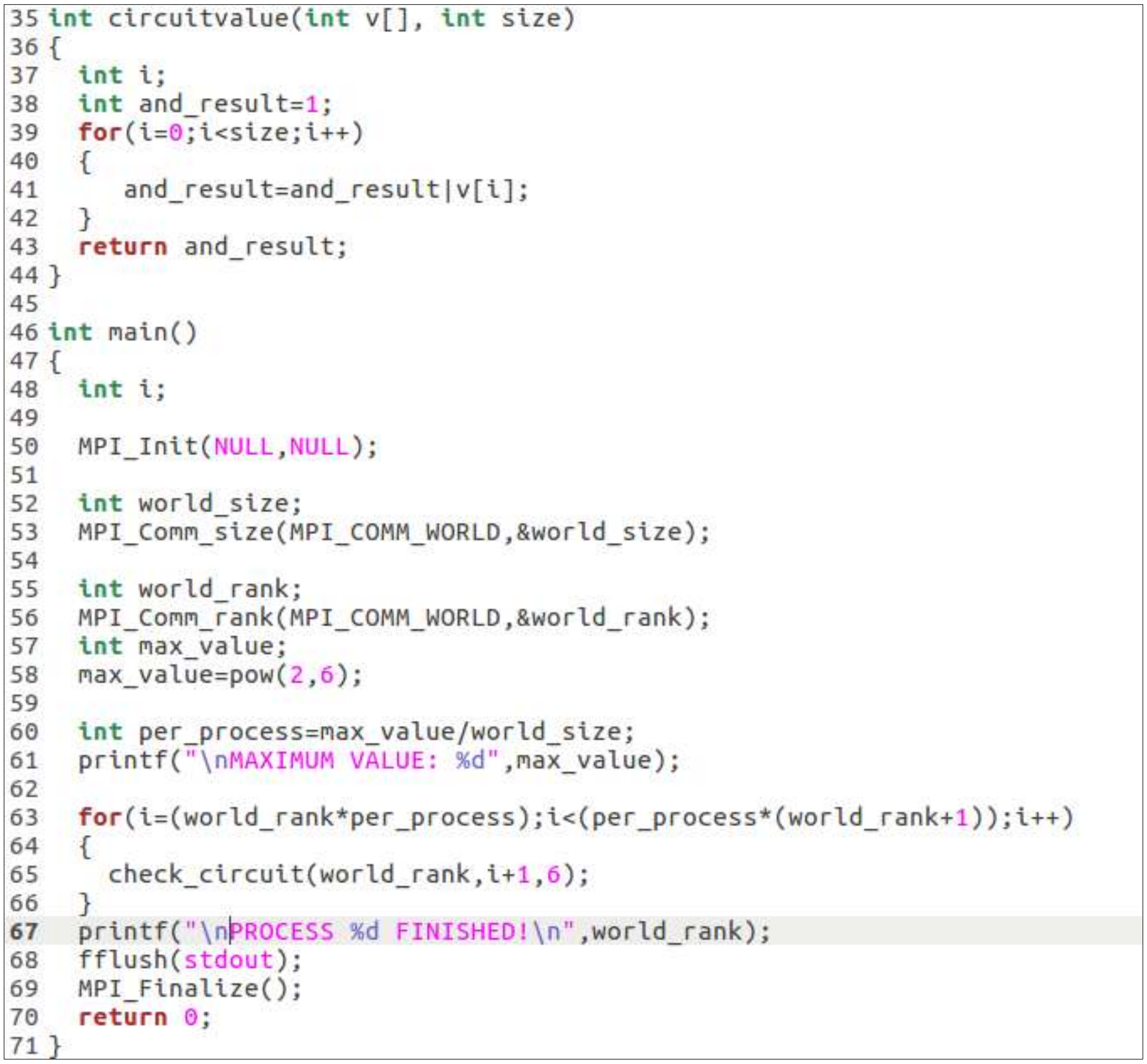
```
000000  0
001000  1
        0
000100  0
        0
001100  0
        0
000010  0
        0
001010  0
        0
000110  0
        0
001110  0
        0
000001  0
        0
111110  0
        0
110001  0
        0
111001  0
        0
110001  0
        0
110101  0
        0
111101  0
        0
110011  0
        0
111011  0
        0
111111  0
        0
dhruv@dhruv-Inspiron-5559:~/PDC-lab/Lab 5/Problem 2$ 
```

Circuit Satisfiability with Number of Solutions

Code

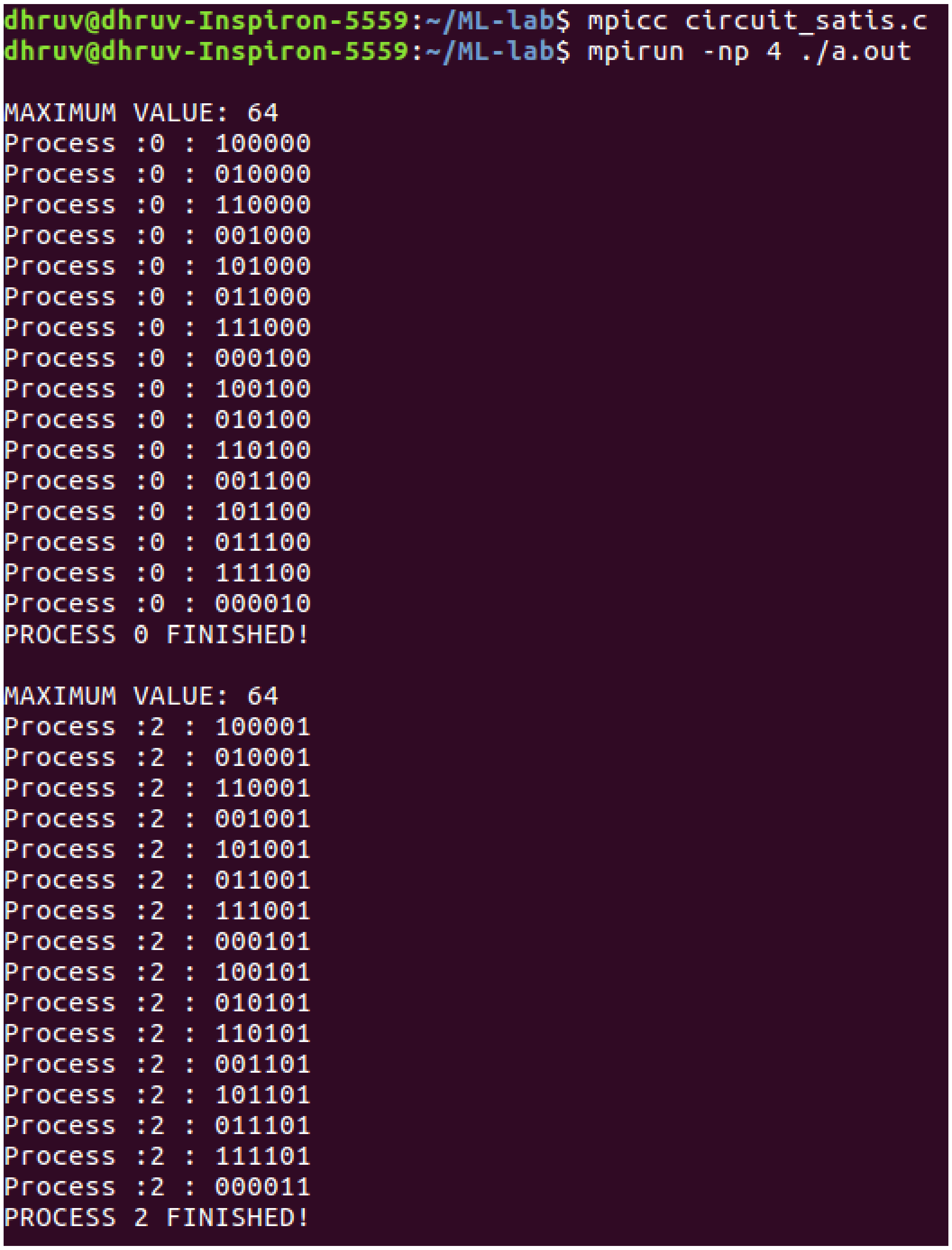


```
#include<mpi.h>
#include<stdio.h>
#include<math.h>
int check(int N,int i);
void check_circuit(int process,int input,int size);
int circuitvalue(int v[],int size);
int check(int N,int i)
{
    if( N & (1 << i) )
        return 1;
    else
        return 0;
}
void check_circuit(int process, int input, int size)
{
  int v[size];
  int i;

  for(i=0;i<size;i++)
  {
    v[i]=check(input,i);
  }

  if(circuitvalue(v,size))
  {
    printf("\nProcess :%d : ",process);
    for(i=0;i<size;i++)
    {
      printf("%d",v[i]);
    }
    //printf("\n");
  }
}

```

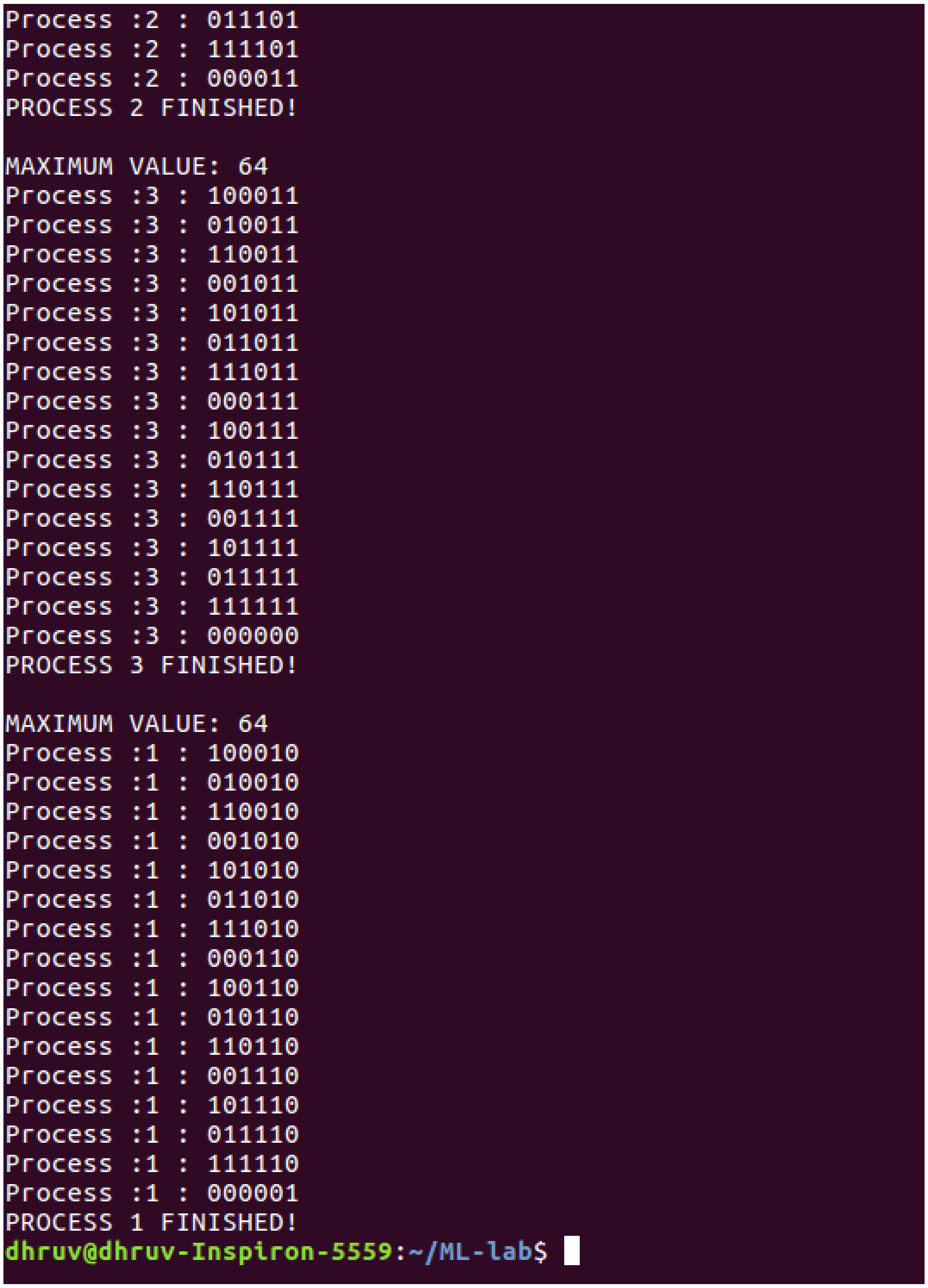


```c
int circuitvalue(int v[], int size)
{
  int i;
  int and_result=1;
  for(i=0;i<size;i++)
  {
     and_result=and_result|v[i];
  }
  return and_result;
}

int main()
{
  int i;

  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);
  int max_value;
  max_value=pow(2,6);

  int per_process=max_value/world_size;
  printf("\nMAXIMUM VALUE: %d",max_value);

  for(i=(world_rank*per_process);i<(per_process*(world_rank+1));i++)
  {
    check_circuit(world_rank,i+1,6);
  }
  printf("\nPROCESS %d FINISHED!\n",world_rank);
  fflush(stdout);
  MPI_Finalize();
  return 0;
}
```

Output



```
dhruv@dhruv-Inspiron-5559:~/ML-lab$ mpicc circuit_satis.c
dhruv@dhruv-Inspiron-5559:~/ML-lab$ mpirun -np 4 ./a.out

MAXIMUM VALUE: 64
Process :0 : 100000
Process :0 : 010000
Process :0 : 110000
Process :0 : 001000
Process :0 : 101000
Process :0 : 011000
Process :0 : 111000
Process :0 : 000100
Process :0 : 100100
Process :0 : 010100
Process :0 : 110100
Process :0 : 001100
Process :0 : 101100
Process :0 : 011100
Process :0 : 111100
Process :0 : 000010
PROCESS 0 FINISHED!

MAXIMUM VALUE: 64
Process :2 : 100001
Process :2 : 010001
Process :2 : 110001
Process :2 : 001001
Process :2 : 101001
Process :2 : 011001
Process :2 : 111001
Process :2 : 000101
Process :2 : 100101
Process :2 : 010101
Process :2 : 110101
Process :2 : 001101
Process :2 : 101101
Process :2 : 011101
Process :2 : 111101
Process :2 : 000011
PROCESS 2 FINISHED!
```



```
Process :2 : 011101
Process :2 : 111101
Process :2 : 000011
PROCESS 2 FINISHED!

MAXIMUM VALUE: 64
Process :3 : 100011
Process :3 : 010011
Process :3 : 110011
Process :3 : 001011
Process :3 : 101011
Process :3 : 011011
Process :3 : 111011
Process :3 : 000111
Process :3 : 100111
Process :3 : 010111
Process :3 : 110111
Process :3 : 001111
Process :3 : 101111
Process :3 : 011111
Process :3 : 111111
Process :3 : 000000
PROCESS 3 FINISHED!

MAXIMUM VALUE: 64
Process :1 : 100010
Process :1 : 010010
Process :1 : 110010
Process :1 : 001010
Process :1 : 101010
Process :1 : 011010
Process :1 : 111010
Process :1 : 000110
Process :1 : 100110
Process :1 : 010110
Process :1 : 110110
Process :1 : 001110
Process :1 : 101110
Process :1 : 011110
Process :1 : 111110
Process :1 : 000001
PROCESS 1 FINISHED!
dhruv@dhruv-Inspiron-5559:~/ML-lab$
```