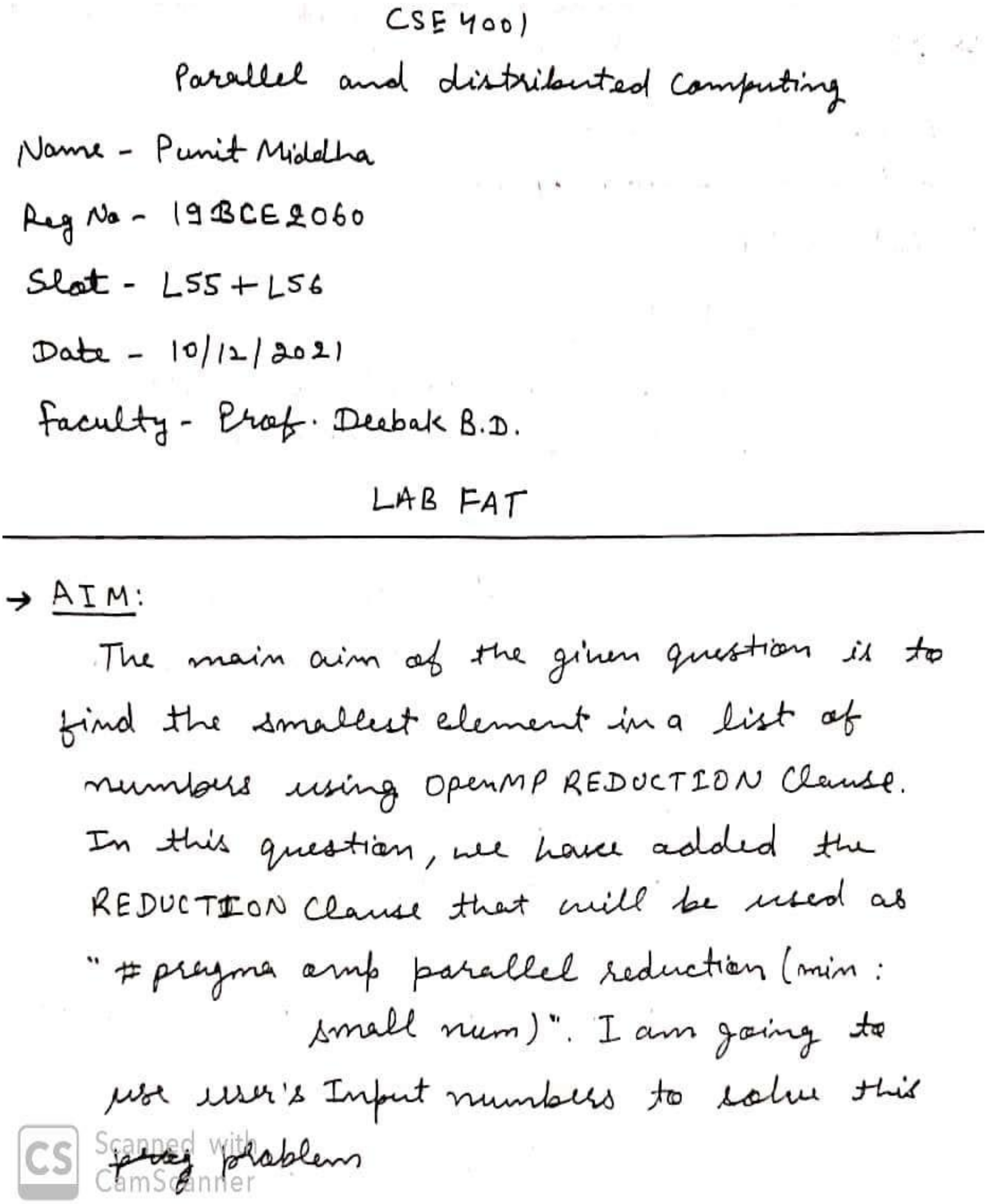
Dated : 10 – 12 – 2021
Assessment No. : FINAL EXAM

School of Computer Science and Engineering
FALL 2021 – 2022
CSE4001: Parallel and Distributed Computing
SLOT: L55+L56

NAME: PUNIT MIDDHA **REGNO: 19BCE2060**

Aim



CSE 4001

Parallel and distributed Computing

Name - Punit Middha

Reg No - 19BCE2060

Slot - L55 + L56

Date - 10/12/2021

faculty - Prof. Deebak B.D.

LAB FAT

→ AIM:

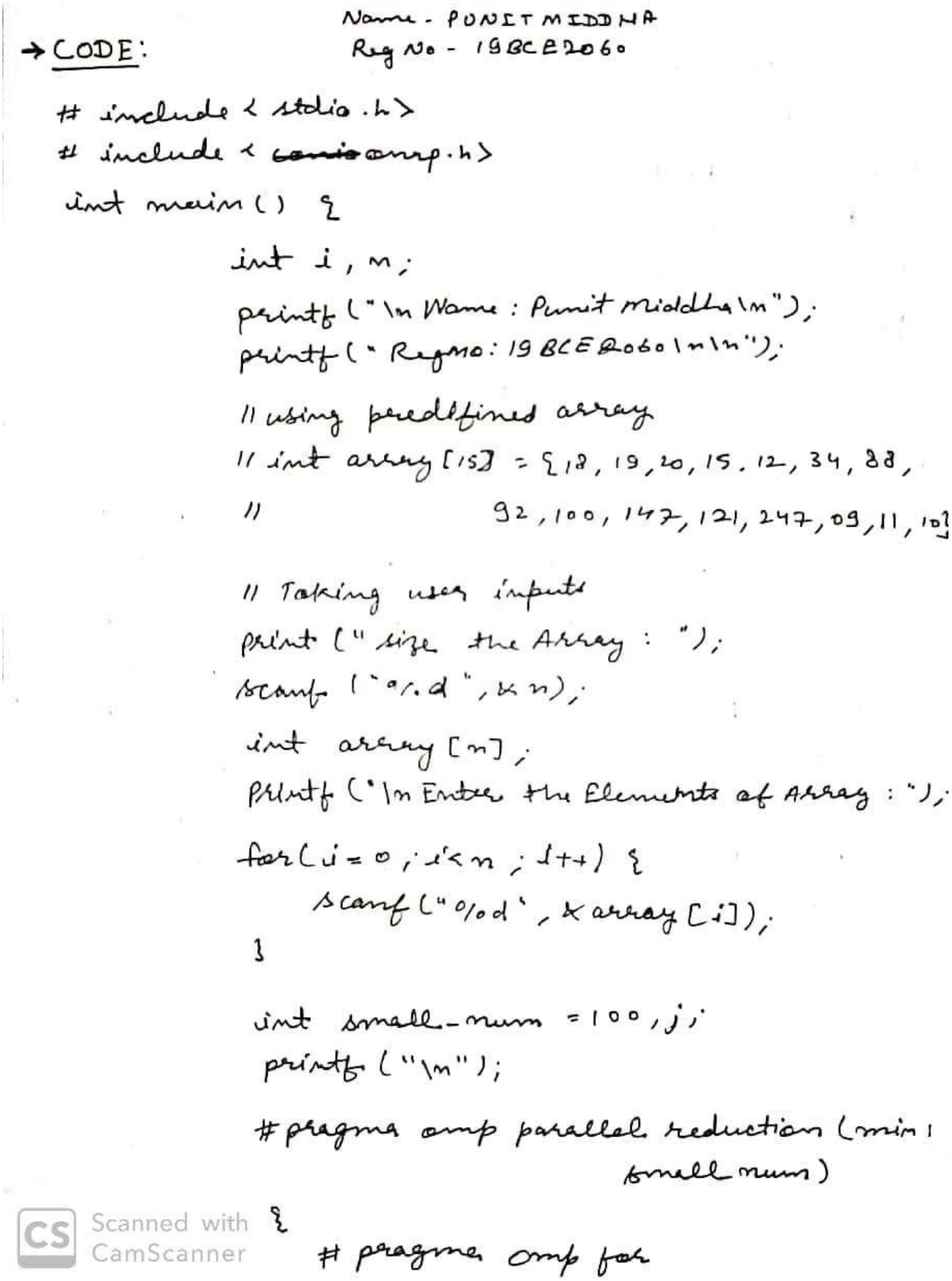
The main aim of the given question is to find the smallest element in a list of numbers using OpenMP REDUCTION Clause. In this question, we have added the REDUCTION Clause that will be used as "#pragma omp parallel reduction (min : small num)". I am going to use user's Input numbers to solve this ~~prog~~ problem

Dated : 10 – 12 – 2021
Assessment No. : FINAL EXAM

School of Computer Science and Engineering
FALL 2021 – 2022
CSE4001: Parallel and Distributed Computing
SLOT: L55+L56

NAME: PUNIT MIDDHA **REGNO: 19BCE2060**

Source Code



Name - PUNIT MIDDHA
Reg No - 19BCE2060

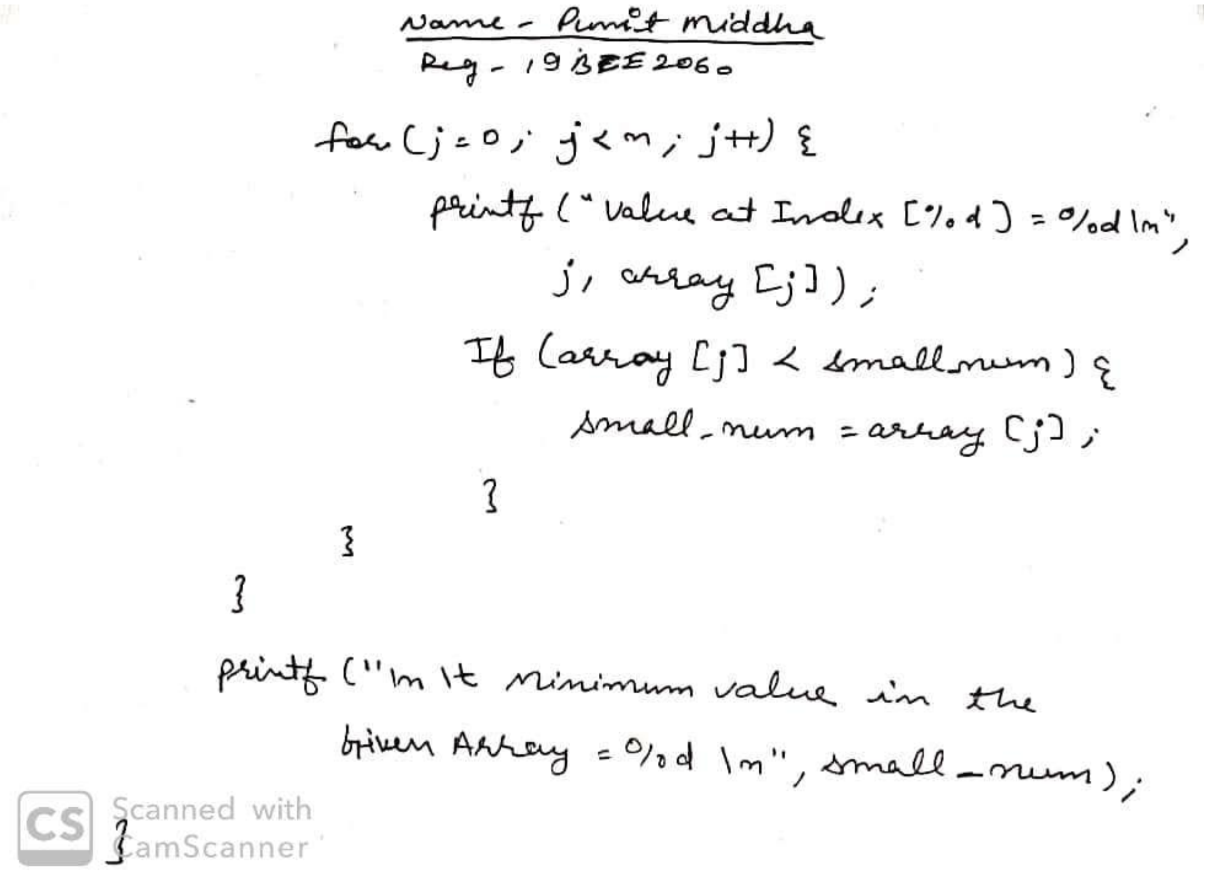
→ CODE:

```
# include < stdio.h>
# include < ~~conio~~omp.h>
int main () {

        int i, n;
        printf ("\n Name : Punit Middha\n");
        printf (" Regno: 19 BCE2060\n\n");

        // using predefined array
        // int array [15] = {18, 19, 20, 15. 12, 34, 88,
        //           92, 100, 147, 121, 247, 09, 11, 10}

        // Taking user inputs
        print (" size the Array : ");
        scanf ("%d", & n);
        int array [n];
        Printf ("\n Enter the Elements of Array : ");

        for (i=0; i<n; i++) {
            scanf ("%d", & array [i]);
        }

        int small_num = 100, j;
        printf ("\n");

        #pragma omp parallel reduction (min :
                                  small num)
        {
            # pragma omp for
```

Dated : 10 – 12 – 2021
Assessment No. : FINAL EXAM

School of Computer Science and Engineering
FALL 2021 – 2022
CSE4001: Parallel and Distributed Computing
SLOT: L55+L56

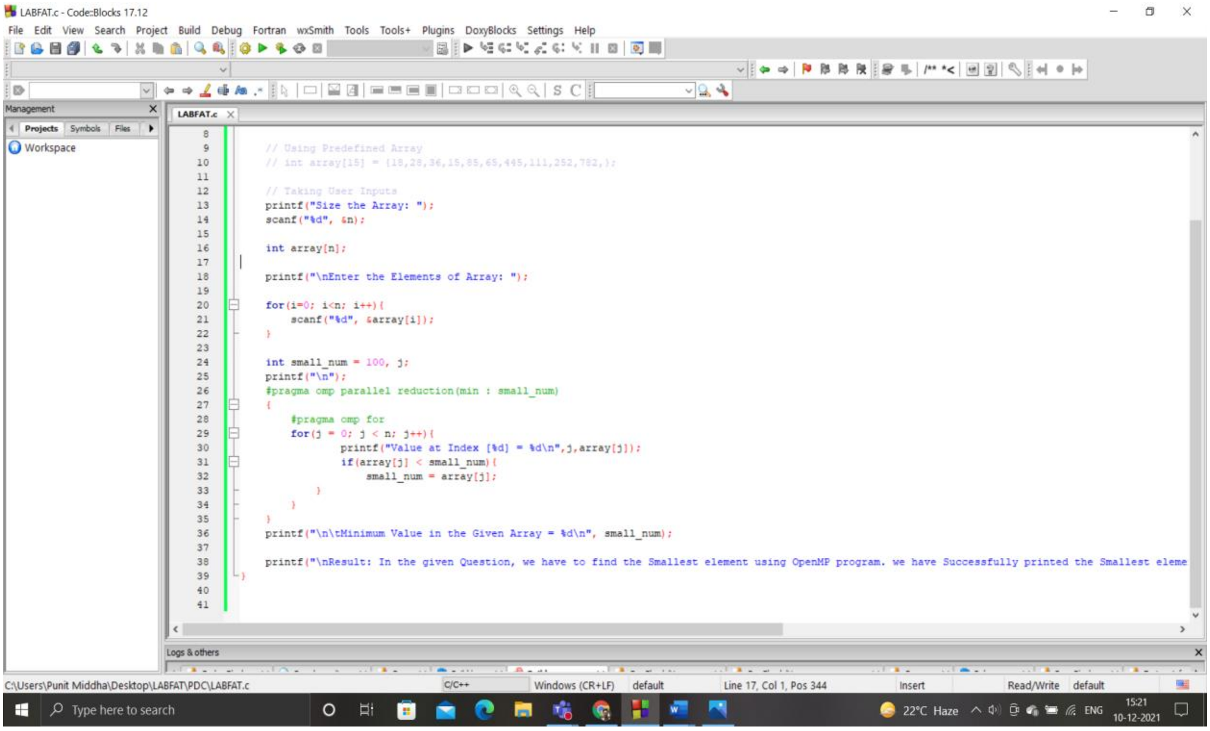
NAME: PUNIT MIDDHA **REGNO: 19BCE2060**



Name - Punit Middha
Reg - 19BCE2060

```
for (j = 0; j < n; j++) {
    printf ("Value at Index [%d] = %d \n",
        j, array [j]);
    If (array [j] < small_num) {
        small_num = array [j];
    }
}
}
printf ("\n \t Minimum value in the
    Given Array = %d \n", small_num);
}
```

Digital Screenshot:



LABFAT.c - Code::Blocks 17.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Management
Projects Symbols Files
Workspace
LABFAT.c
Logs & others
C:\Users\Punit Middha\Desktop\LABFAT\PDC\LABFAT.c
C/C++
Windows (CR+LF)
default
Line 17, Col 1, Pos 344
Insert
Read/Write
default
Type here to search
22°C Haze
ENG
15:21
10-12-2021

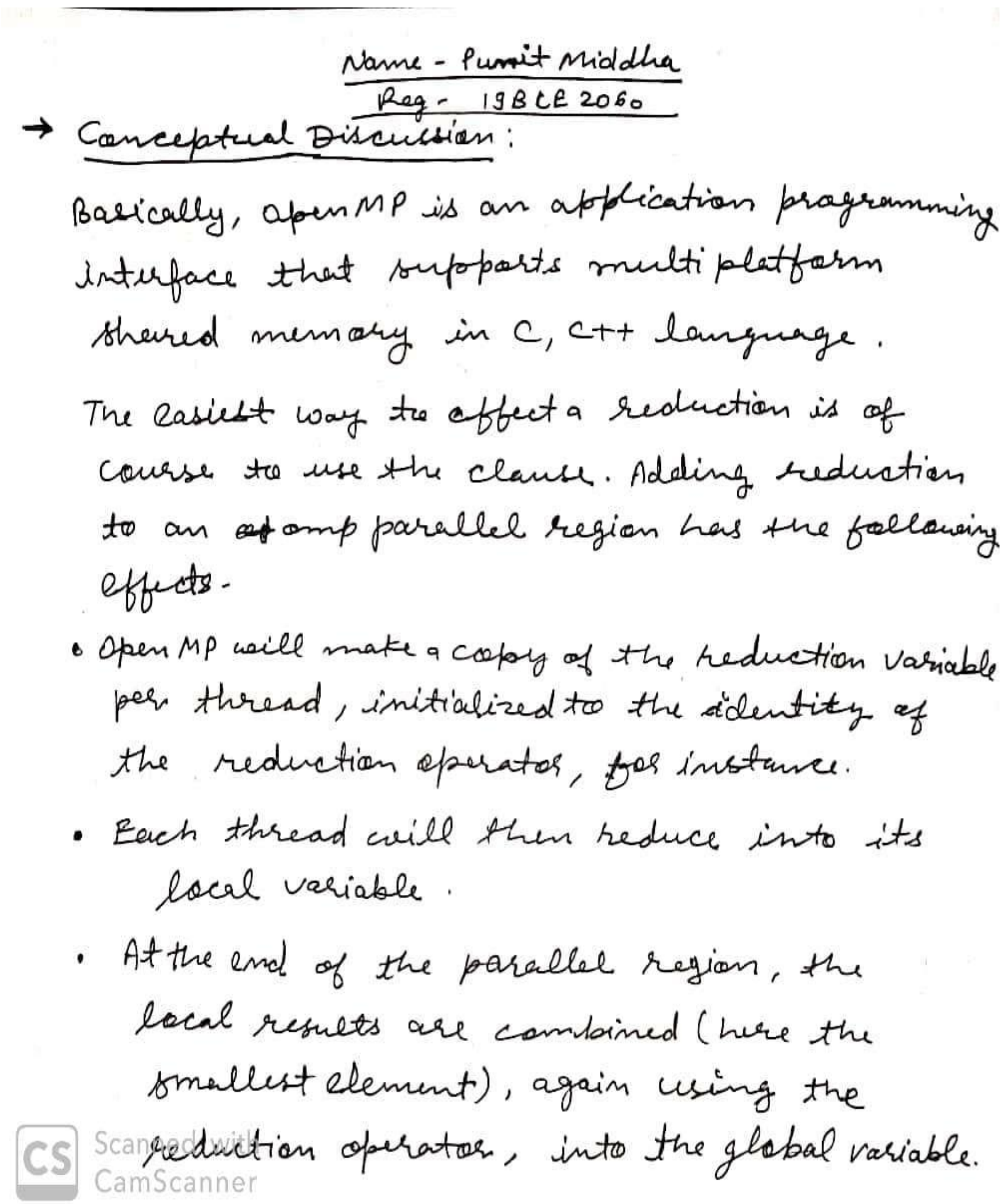
```

    // Using Predefined Array
    // int array[15] = {18,28,36,15,85,65,445,111,252,782,};

    // Taking User Inputs
    printf("Size the Array: ");
    scanf("%d", &n);

    int array[n];

    printf("\nEnter the Elements of Array: ");

    for(i=0; i<n; i++){
        scanf("%d", &array[i]);
    }

    int small_num = 100, j;
    printf("\n");
    #pragma omp parallel reduction(min : small_num)
    {
        #pragma omp for
        for(j = 0; j < n; j++){
                printf("Value at Index [%d] = %d\n",j,array[j]);
                if(array[j] < small_num){
                    small_num = array[j];
                }
        }
    }
    printf("\n\tMinimum Value in the Given Array = %d\n", small_num);

    printf("\nResult: In the given Question, we have to find the Smallest element using OpenMP program. we have Successfully printed the Smallest eleme
}


```

Dated : 10 – 12 – 2021
Assessment No. : FINAL EXAM

School of Computer Science and Engineering
FALL 2021 – 2022
CSE4001: Parallel and Distributed Computing
SLOT: L55+L56

NAME: PUNIT MIDDHA **REGNO: 19BCE2060**

Conceptual Discussion



Name - Punit Middha
Reg - 19BCE2060

→ Conceptual Discussion:

Basically, openMP is an application programming interface that supports multiplatform shared memory in C, C++ language.

The easiest way to affect a reduction is of course to use the clause. Adding reduction to an ~~of~~ omp parallel region has the following effects-

- OpenMP will make a copy of the reduction variable per thread, initialized to the identity of the reduction operator, for instance.
- Each thread will then reduce into its local variable.
- At the end of the parallel region, the local results are combined (here the smallest element), again using the reduction operator, into the global variable.

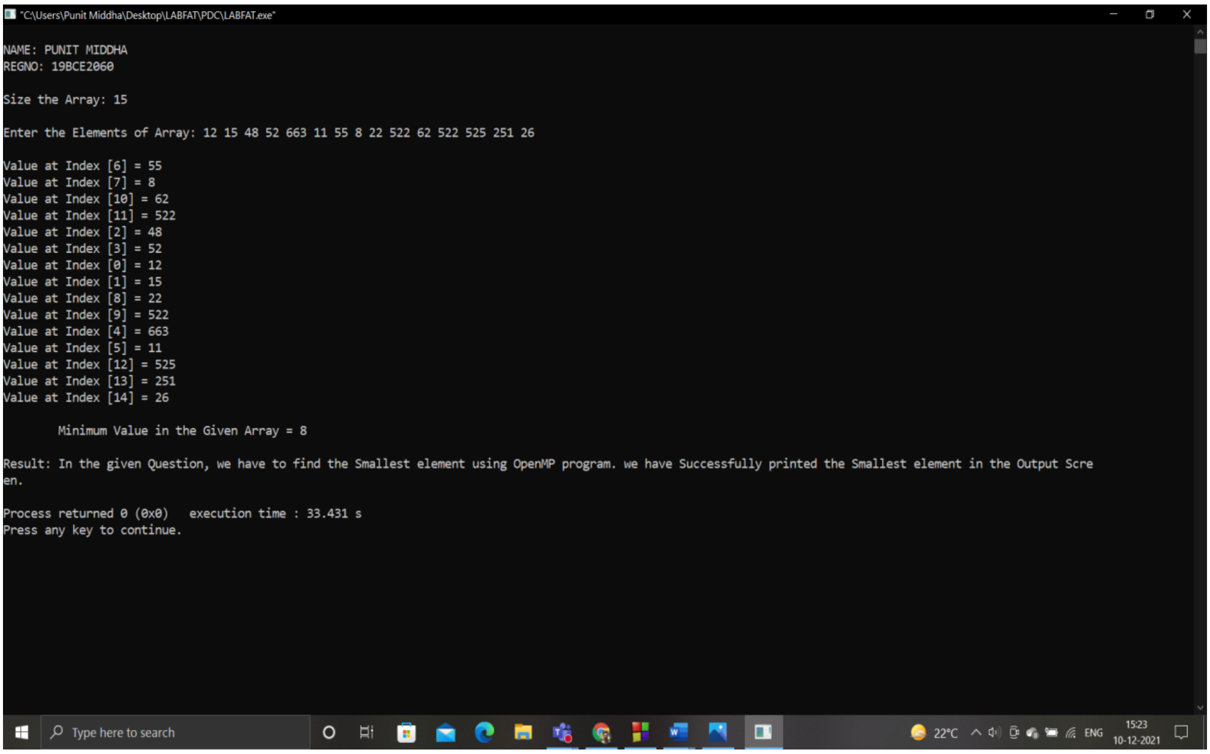
Dated : 10 – 12 – 2021
Assessment No. : FINAL EXAM

School of Computer Science and Engineering
FALL 2021 – 2022
CSE4001: Parallel and Distributed Computing
SLOT: L55+L56

NAME: PUNIT MIDDHA **REGNO: 19BCE2060**

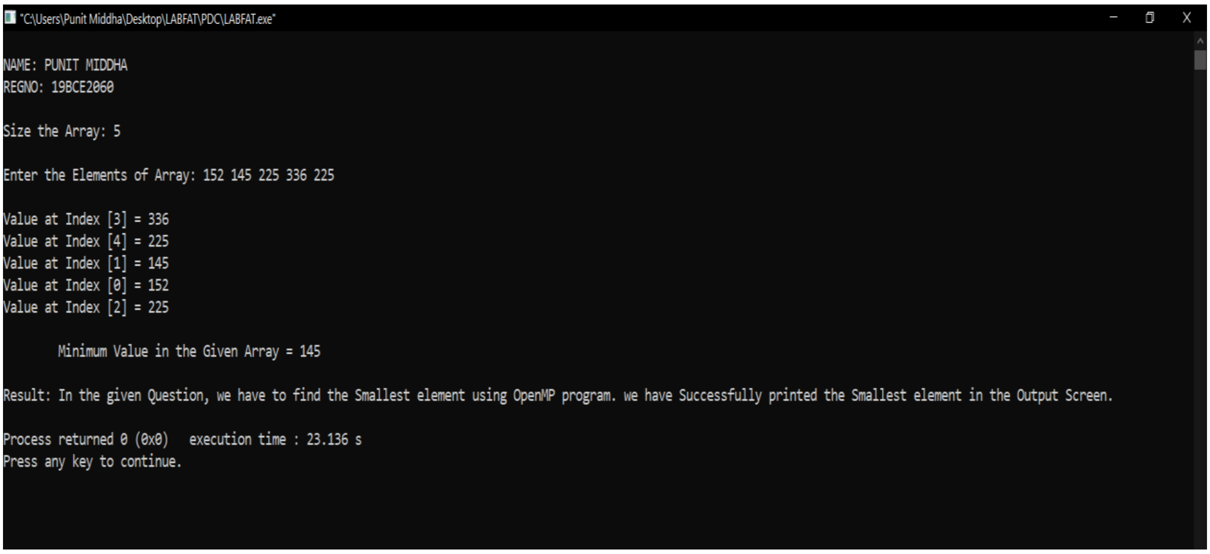
Execution Output

Test case 1:



```
NAME: PUNIT MIDDHA
REGNO: 19BCE2060

Size the Array: 15

Enter the Elements of Array: 12 15 48 52 663 11 55 8 22 522 62 522 525 251 26

Value at Index [6] = 55
Value at Index [7] = 8
Value at Index [10] = 62
Value at Index [11] = 522
Value at Index [2] = 48
Value at Index [3] = 52
Value at Index [0] = 12
Value at Index [1] = 15
Value at Index [8] = 22
Value at Index [9] = 522
Value at Index [4] = 663
Value at Index [5] = 11
Value at Index [12] = 525
Value at Index [13] = 251
Value at Index [14] = 26

        Minimum Value in the Given Array = 8

Result: In the given Question, we have to find the Smallest element using OpenMP program. we have Successfully printed the Smallest element in the Output Scre
en.

Process returned 0 (0x0)   execution time : 33.431 s
Press any key to continue.
```

Test case 2:



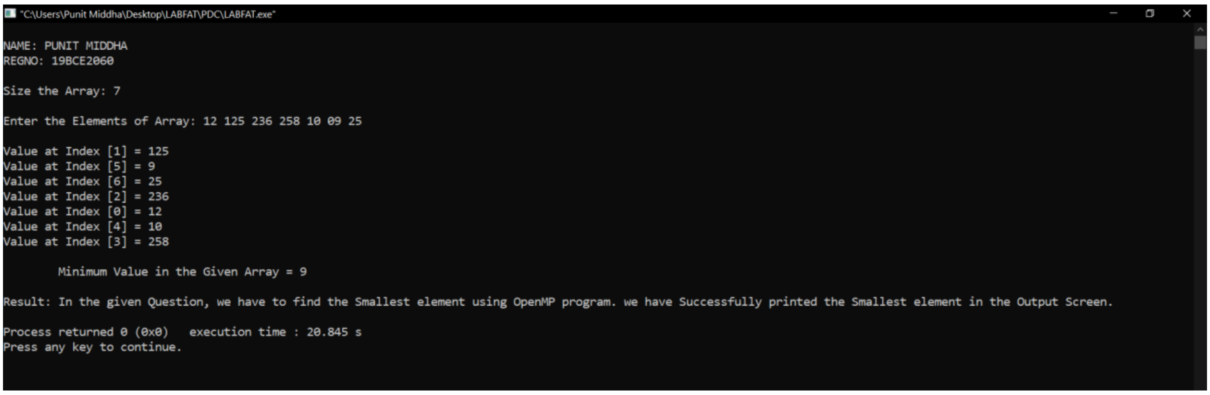
```
NAME: PUNIT MIDDHA
REGNO: 19BCE2060

Size the Array: 5

Enter the Elements of Array: 152 145 225 336 225

Value at Index [3] = 336
Value at Index [4] = 225
Value at Index [1] = 145
Value at Index [0] = 152
Value at Index [2] = 225

        Minimum Value in the Given Array = 145

Result: In the given Question, we have to find the Smallest element using OpenMP program. we have Successfully printed the Smallest element in the Output Screen.

Process returned 0 (0x0)   execution time : 23.136 s
Press any key to continue.
```

Dated : 10 – 12 – 2021
Assessment No. : FINAL EXAM

School of Computer Science and Engineering
FALL 2021 – 2022
CSE4001: Parallel and Distributed Computing
SLOT: L55+L56

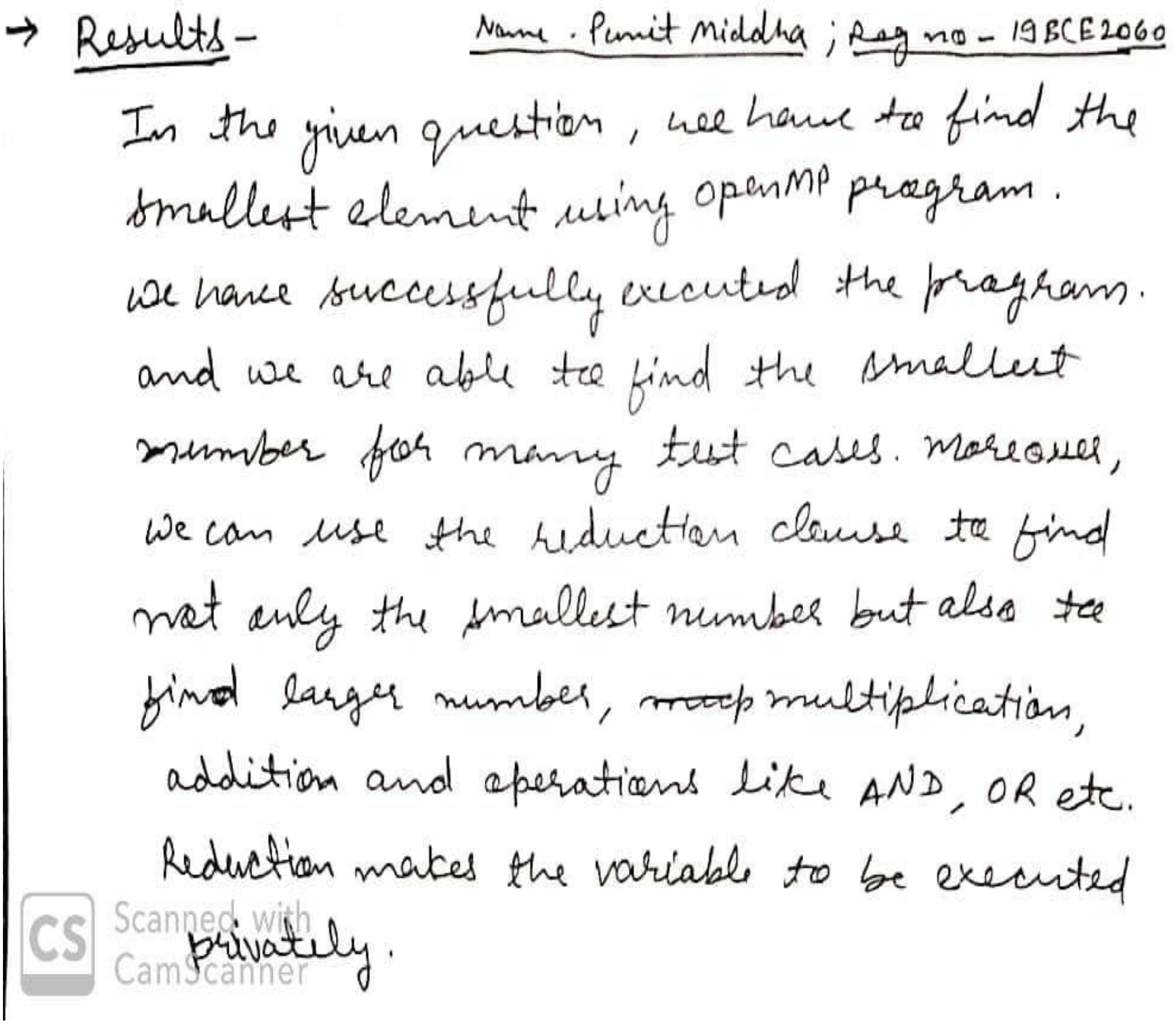
NAME: PUNIT MIDDHA **REGNO: 19BCE2060**

Test case 3:



```
"C:\Users\Punit Middha\Desktop\LABFAT\PDC\LABFAT.exe"

NAME: PUNIT MIDDHA
REGNO: 19BCE2060

Size the Array: 7

Enter the Elements of Array: 12 125 236 258 10 09 25

Value at Index [1] = 125
Value at Index [5] = 9
Value at Index [6] = 25
Value at Index [2] = 236
Value at Index [0] = 12
Value at Index [4] = 10
Value at Index [3] = 258

        Minimum Value in the Given Array = 9

Result: In the given Question, we have to find the Smallest element using OpenMP program. we have Successfully printed the Smallest element in the Output Screen.

Process returned 0 (0x0)   execution time : 20.845 s
Press any key to continue.
```

Results



→ Results – Name . Punit Middha ; Reg no – 19BCE2060

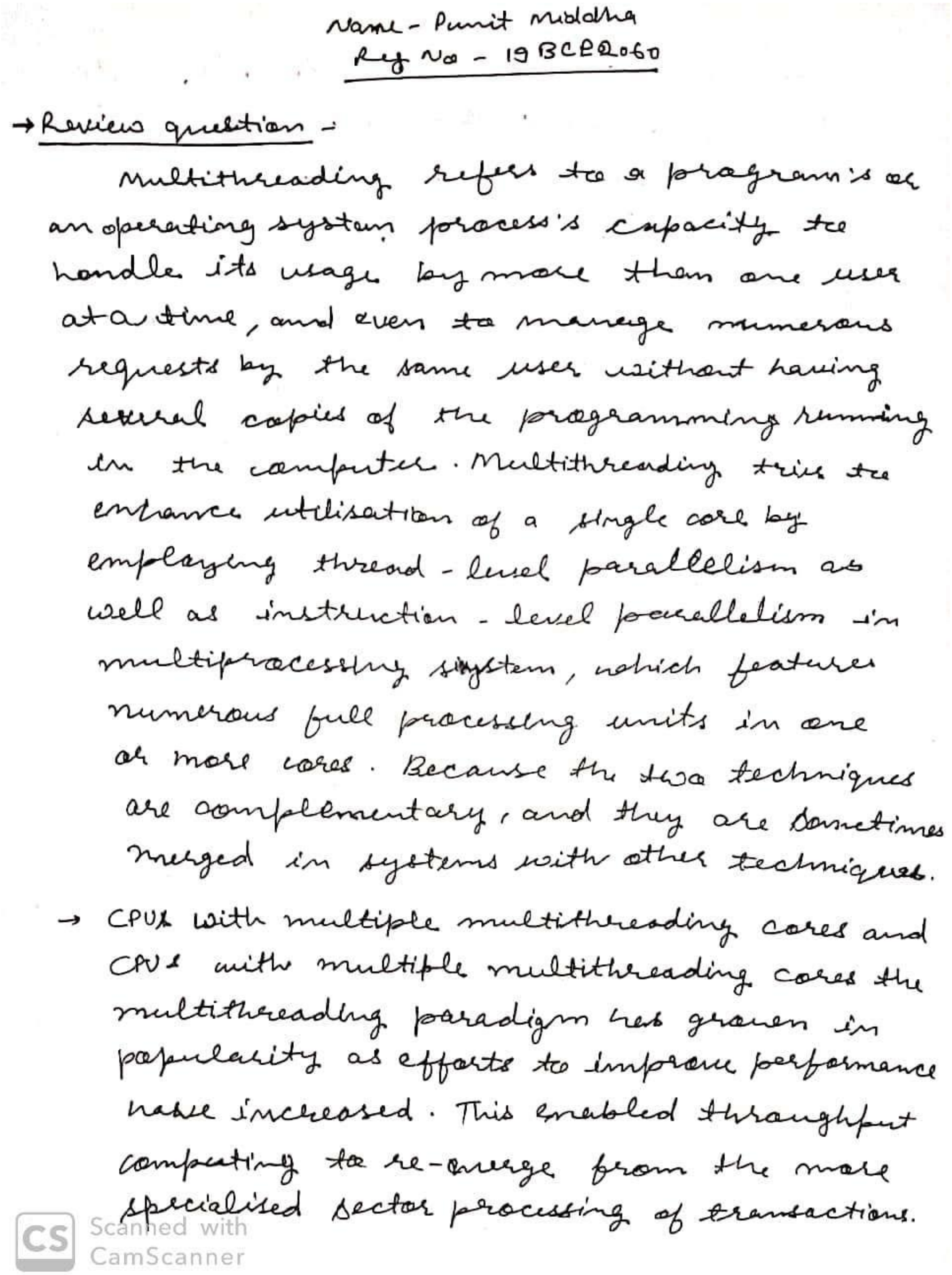
In the given question, we have to find the smallest element using openMP program. We have successfully executed the program. and we are able to find the smallest number for many test cases. Moreover, we can use the reduction clause to find not only the smallest number but also to find larger number, multiplication, addition and operations like AND, OR etc. Reduction makes the variable to be executed privately.

Dated : 10 – 12 – 2021
Assessment No. : FINAL EXAM

School of Computer Science and Engineering
FALL 2021 – 2022
CSE4001: Parallel and Distributed Computing
SLOT: L55+L56

NAME: PUNIT MIDDHA **REGNO: 19BCE2060**

Review Question: [Viva-Voce]



Name - Punit Middha
Reg No - 19BCE2060

→ Review question -

Multithreading refers to a program's or an operating system process's capacity to handle its usage by more than one user at a time, and even to manage numerous requests by the same user without having several copies of the programming running in the computer. Multithreading tries to enhance utilisation of a single core by employing thread-level parallelism as well as instruction-level parallelism in multiprocessing system, which features numerous full processing units in one or more cores. Because the two techniques are complementary, and they are sometimes merged in systems with other techniques.

→ CPUs with multiple multithreading cores and CPUs with multiple multithreading cores the multithreading paradigm has grown in popularity as efforts to improve performance have increased. This enabled throughput computing to re-emerge from the more specialised sector processing of transactions.

Dated : 10 – 12 – 2021
Assessment No. : FINAL EXAM

School of Computer Science and Engineering
FALL 2021 – 2022
CSE4001: Parallel and Distributed Computing
SLOT: L55+L56

NAME: PUNIT MIDDHA **REGNO: 19BCE2060**

Despite the fact that it is extremely difficult to accelerate most computer systems are multitasking, whether it be a single thread or a single application.

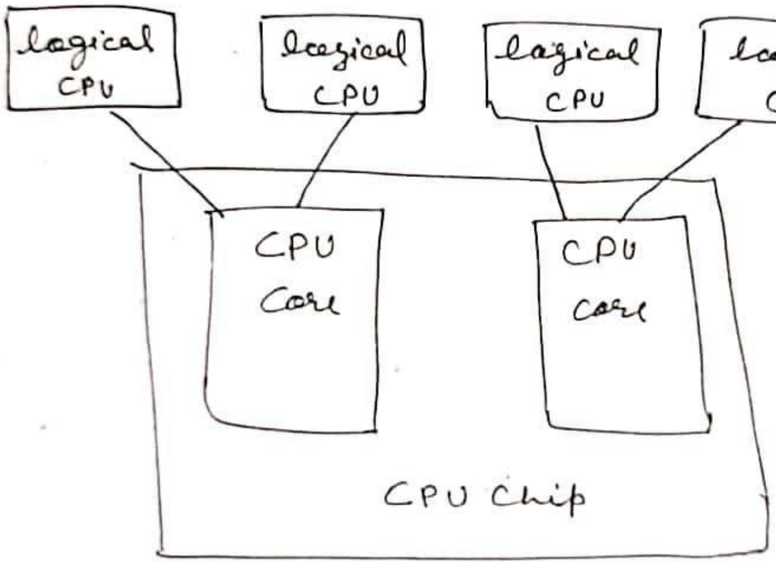


Fig- architecture diagram