|  |  |
|---|---|
| **Name: Varun Menon** | **Course Code: CSE4001** |
| **Reg No: 19BCE1438** | **Faculty: Dr. Harini S** |

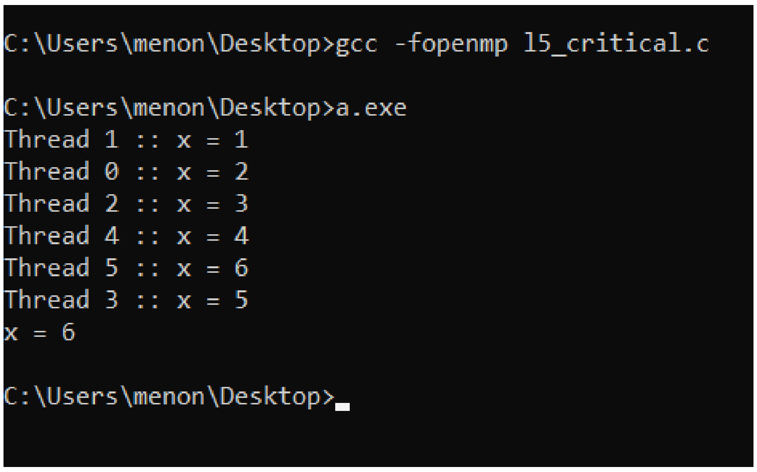
**Lab Experiment 5**

1. Example Program -- Critical

   Code:

```c
#include <stdio.h>
#include <omp.h>
int main()
{
    int x = 0;
    #pragma omp parallel
    {
        #pragma omp critical
        {
            x = x + 1;
        }
        printf("Thread %d :: x = %d\n", omp_get_thread_num(), x);
    }
    printf("x = %d\n", x);
    return 0;
}
```

   Output:



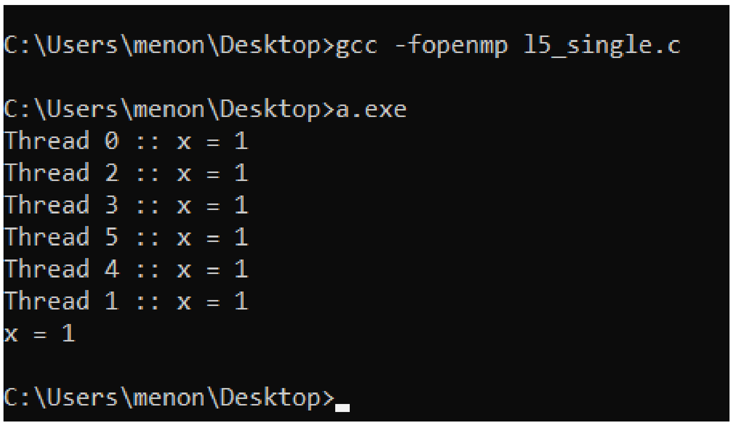
```
C:\Users\menon\Desktop>gcc -fopenmp l5_critical.c

C:\Users\menon\Desktop>a.exe
Thread 1 :: x = 1
Thread 0 :: x = 2
Thread 2 :: x = 3
Thread 4 :: x = 4
Thread 5 :: x = 6
Thread 3 :: x = 5
x = 6

C:\Users\menon\Desktop>
```

2. Example Program -- Single

Code:

```c
#include <stdio.h>
#include <omp.h>
int main()
{
    int x = 0;
    #pragma omp parallel
    {
        #pragma omp single
        {
            x = x + 1;
        }
        printf("Thread %d :: x = %d\n", omp_get_thread_num(), x);
    }
    printf("x = %d\n", x);
    return 0;
}
```

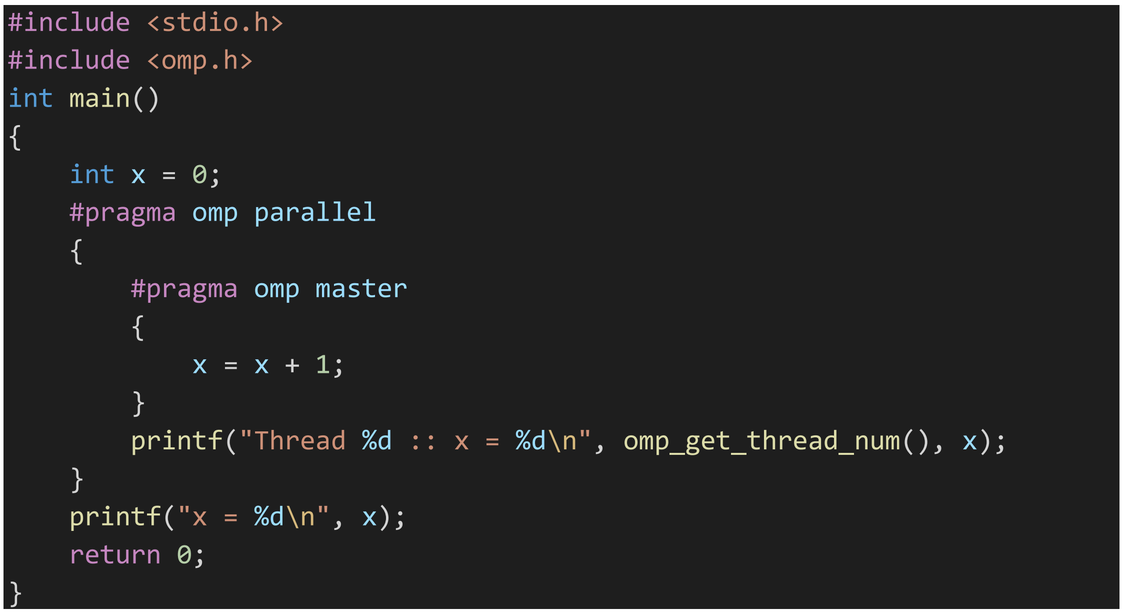
Output:



```
C:\Users\menon\Desktop>gcc -fopenmp 15_single.c

C:\Users\menon\Desktop>a.exe
Thread 0 :: x = 1
Thread 2 :: x = 1
Thread 3 :: x = 1
Thread 5 :: x = 1
Thread 4 :: x = 1
Thread 1 :: x = 1
x = 1

C:\Users\menon\Desktop>
```

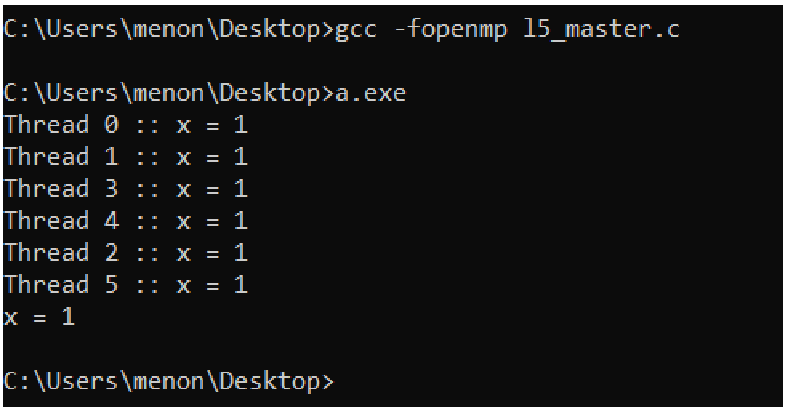
3. Example Program -- Master

Code:



```c
#include <stdio.h>
#include <omp.h>
int main()
{
    int x = 0;
    #pragma omp parallel
    {
        #pragma omp master
        {
            x = x + 1;
        }
        printf("Thread %d :: x = %d\n", omp_get_thread_num(), x);
    }
    printf("x = %d\n", x);
    return 0;
}
```

Output:



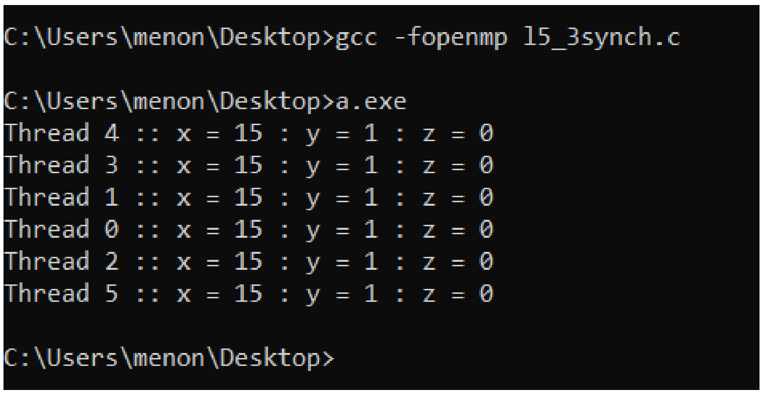
```
C:\Users\menon\Desktop>gcc -fopenmp 15_master.c

C:\Users\menon\Desktop>a.exe
Thread 0 :: x = 1
Thread 1 :: x = 1
Thread 3 :: x = 1
Thread 4 :: x = 1
Thread 2 :: x = 1
Thread 5 :: x = 1
x = 1

C:\Users\menon\Desktop>
```

4. Example program with x=x+thread_id for critical, single and master. To prove concept, use one shared variable for each synchronization construct.

   Code:

```c
#include <stdio.h>
#include <omp.h>
int main()
{
    int x = 0, y = 0, z = 0;
    #pragma omp parallel shared(x, y, z)
    {
        #pragma omp critical
        {
            x = x + omp_get_thread_num();
        }
        #pragma omp single
        {
            y = y + omp_get_thread_num();
        }
        #pragma omp master
        {
            z = z + omp_get_thread_num();
        }
        printf("Thread %d :: x = %d : y = %d : z = %d\n",
omp_get_thread_num(), x, y, z);
    }
    return 0;
}
```

Output:



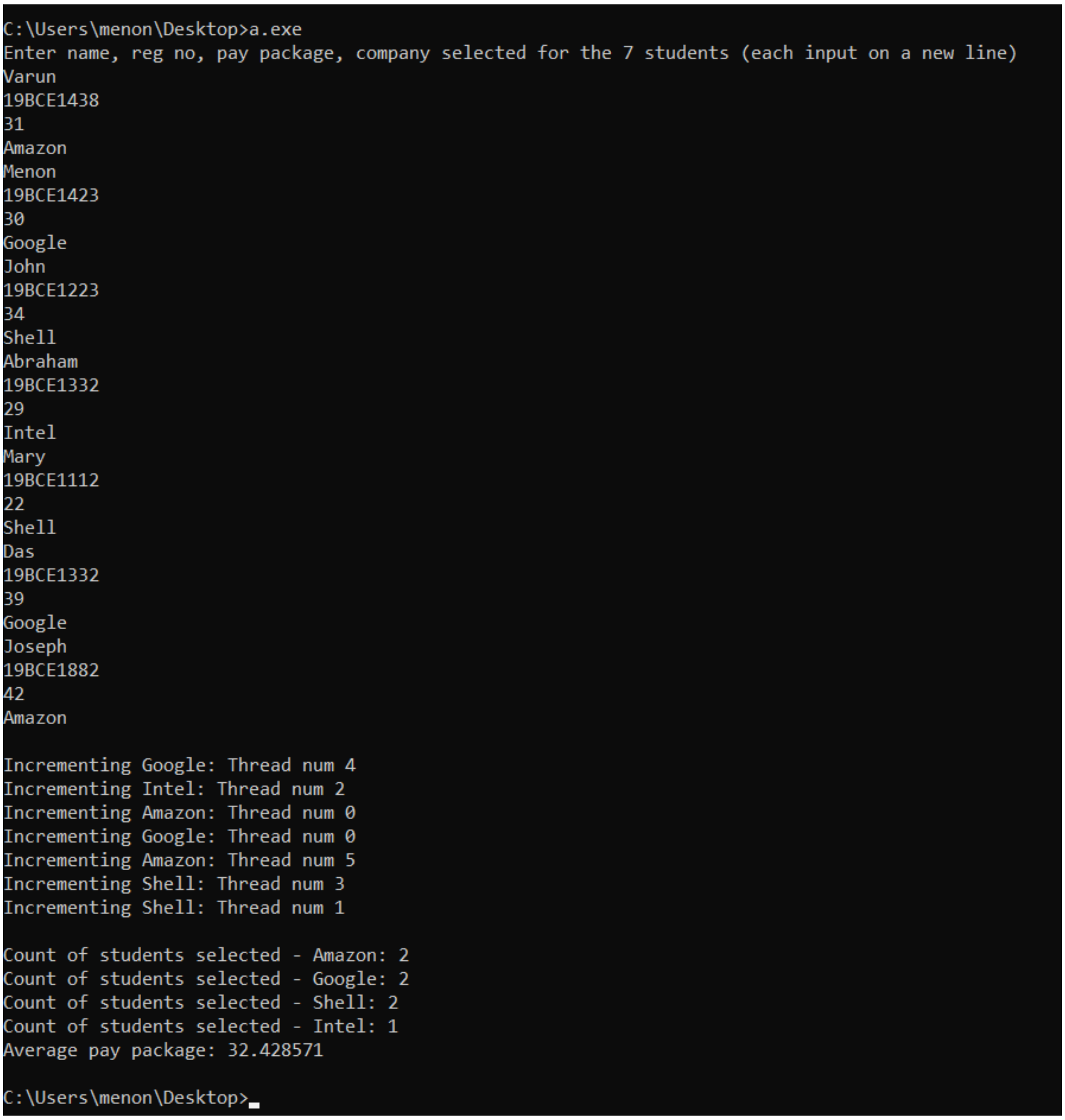
```
C:\Users\menon\Desktop>gcc -fopenmp l5_3synch.c

C:\Users\menon\Desktop>a.exe
Thread 4 :: x = 15 : y = 1 : z = 0
Thread 3 :: x = 15 : y = 1 : z = 0
Thread 1 :: x = 15 : y = 1 : z = 0
Thread 0 :: x = 15 : y = 1 : z = 0
Thread 2 :: x = 15 : y = 1 : z = 0
Thread 5 :: x = 15 : y = 1 : z = 0

C:\Users\menon\Desktop>
```

5. Consider you have to write a program for VIT placement cell where 100 students are placed in 4 companies namely, Amazon, Google, Shell, and Intel. Assume no student is offered more than one placement offer. The program has to do the following tasks in parallel and display the result with thread id.
    - Get as input the name, register number, the pay package of students selected for jobs in the particular organization
    - Display the total number of students selected in each company.
    - Display the average pay package of the 100 students.

Code:

```
#include <stdio.h>
#include <omp.h>
#include<string.h>
#include<stdlib.h>
int main()
{
    int n = 100;
    char names[n][30];
    char reg_nos[n][9];
    int packages[n];
    char companies[n][10];
    //char amazon[] = "Amazon", google[] = "Google", shell[] = "Shell",
intel[] = "Intel";
    //printf("%s %s %s %s\n", google, amazon, shell, intel);
    printf("Enter name, reg no, pay package, company selected for the
%d students (each input on a new line)\n", n);
    for(int i = 0; i < n; i++)
    {
        scanf("%s", names[i]);
        scanf("%s", reg_nos[i]);
        scanf("%d", &packages[i]);
        scanf("%s", companies[i]);
    }
    int cnt_amazon = 0, cnt_google = 0, cnt_shell = 0, cnt_intel = 0;
    double average_pay = 0.0f;
    printf("\n");
    #pragma omp parallel for shared(packages, companies, cnt_amazon,
cnt_google, cnt_shell, cnt_intel, average_pay)
    for(int i = 0; i < n; i++)
    {
        #pragma omp critical
        if(strcmp("Amazon", companies[i]) == 0)
        {
            printf("Incrementing Amazon: Thread num %d\n",
omp_get_thread_num());
            cnt_amazon++;
        }
        #pragma omp critical
        if(strcmp("Google", companies[i]) == 0)
        {
            printf("Incrementing Google: Thread num %d\n",
omp_get_thread_num());
            cnt_google++;
        }
        #pragma omp critical
        if(strcmp("Shell", companies[i]) == 0)
        {
            printf("Incrementing Shell: Thread num %d\n",
omp_get_thread_num());
            cnt_shell++;
        }
        #pragma omp critical
        if(strcmp("Intel", companies[i]) == 0)
        {
            printf("Incrementing Intel: Thread num %d\n",
omp_get_thread_num());
            cnt_intel++;
        }
        #pragma omp critical
        average_pay += 1.0f*packages[i]/n;
    }
    printf("\n");
    printf("Count of students selected - Amazon: %d\n", cnt_amazon);
    printf("Count of students selected - Google: %d\n", cnt_google);
    printf("Count of students selected - Shell: %d\n", cnt_shell);
    printf("Count of students selected - Intel: %d\n", cnt_intel);
    printf("Average pay package: %f\n", average_pay);
    return 0;
}
```

Output:
For the sake of ease of demonstration, we take the number of students as 7 for this output image



```
C:\Users\menon\Desktop>a.exe
Enter name, reg no, pay package, company selected for the 7 students (each input on a new line)
Varun
19BCE1438
31
Amazon
Menon
19BCE1423
30
Google
John
19BCE1223
34
Shell
Abraham
19BCE1332
29
Intel
Mary
19BCE1112
22
Shell
Das
19BCE1332
39
Google
Joseph
19BCE1882
42
Amazon

Incrementing Google: Thread num 4
Incrementing Intel: Thread num 2
Incrementing Amazon: Thread num 0
Incrementing Google: Thread num 0
Incrementing Amazon: Thread num 5
Incrementing Shell: Thread num 3
Incrementing Shell: Thread num 1

Count of students selected - Amazon: 2
Count of students selected - Google: 2
Count of students selected - Shell: 2
Count of students selected - Intel: 1
Average pay package: 32.428571

C:\Users\menon\Desktop>
```