NAME – ANSH GOEL

REGISTER NO.- 20BCE1798

PARALLEL AND DISTRIBUTED COMPUTING

LAB – 5

Q1)Write your own code snippet to demonstrate the following

a)Barrier

Code:

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

OUTPUT:

```
ansh@ansh:~$ gedit pdclab5i.c
ansh@ansh:~$ gcc -o a pdclab5i.c -fopenmp
ansh@ansh:~$ ./a
1
2
1
```

b)MASTER
Code:
#include<omp.h>

```
#include<stdio.h>

int main()

{

int x;

x=0;

#pragma omp parallel shared(x)

{

#pragma omp master

{ x=x+1;

printf("%d \n",x);

}

}

return 0;

}
```

OUTPUT:

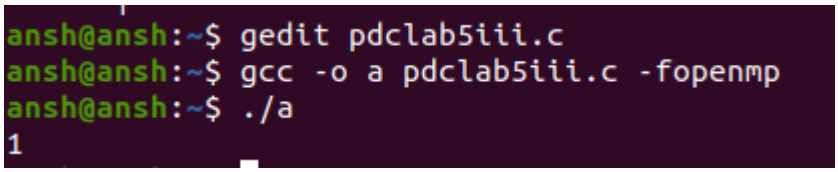
```
ansh@ansh:~$ gedit pdclab5ii.c
ansh@ansh:~$ gcc -o a pdclab5ii.c -fopenmp
ansh@ansh:~$ ./a
1
```

c)SINGLE
Code:
#include<omp.h>

```
#include<omp.h>

#include<stdio.h>

int main()

{
```

```
int x;
x=0;
#pragma omp parallel shared(x)
{
#pragma omp single
{
int tid= omp_get_thread_num();
x=x+tid;
printf("%d \n",x);
}
}
return 0;
}
```

OUTPUT:



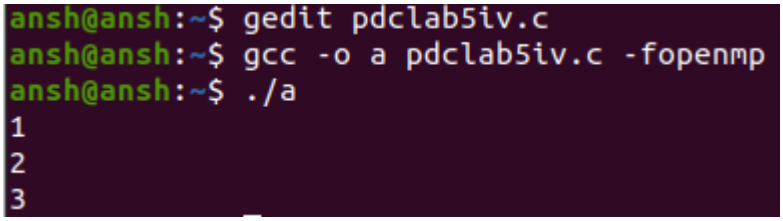
d)CRITICAL

Code:

```
#include<omp.h>
#include<stdio.h>
int main()
{
int x;
x=0;
#pragma omp parallel shared(x)
{
#pragma omp critical
{ x=x+l;
printf("%d \n",x);
```

```
}
}
return 0;
}
```

OUTPUT:



e)ORDERED
Code:
OUTPUT:

Q2)Write a parallel program in OpenMP API to implement readers-writers problem. Ensure the usage of suitable low-level synchronization constructs. Document the result of synchronization overhead incurred in your experiment.

Code:

```
#include<stdio.h>
#include<omp.h>
#include<string.h>

int main()
{
    char buf1[100];
    strcpy(buf1,"hi");
    int i;
    omp_set_num_threads(2);
```

```
    omp_lock_t writelock;

    omp_init_lock(&writelock);

    #pragma omp parallel
    {
        int tid=omp_get_thread_num();
        if(tid==0)
        {
            printf("I am THREAD:%d
writing",omp_get_thread_num());

            omp_set_lock(&writelock);
            strcat(buf1," writing to the
buffer\n");
            omp_unset_lock(&writelock);
        }
        //strcat(buf1,"contents are getting
written");
        if(tid==1)
        {
            omp_set_lock(&writelock);
            printf("\nI am THREAD:%d
reading",omp_get_thread_num());
            printf("\nthe buffer contents
are:\n %s",buf1);
```

```
            omp_unset_lock(&writelock);
        }


    }
    return 0;
}
```

OUTPUT: