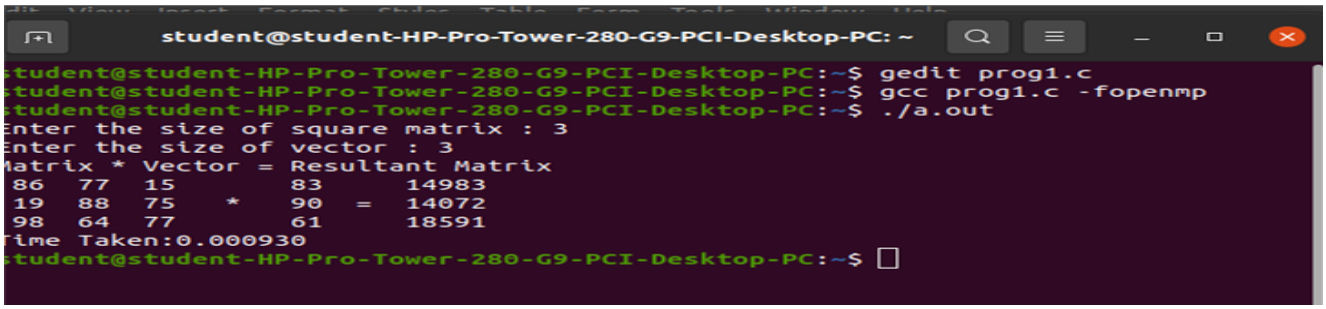
HPC LAB PROGRAMS

PROGRAM1:

```
#include<stdio.h>
#include<stdlib.h>
#include<omp.h>
#include<time.h>
void main() {
	int m,n;
	printf("Enter the size of square matrix : ");
	scanf("%d",&n);
	printf("Enter the size of vector : ");
	scanf("%d", &m);
	time_t st,et;
	st=clock();
	if (m!=n) {
		printf("Multiplication is not possible.\n");
		exit(0);
	}
	int i=0,j=0;
	int **arr=(int**)malloc(n*sizeof(int*));
	int *vec=(int*)malloc(n*sizeof(int));
	int *res=(int*)malloc(n*sizeof(int));
	omp_set_num_threads(n);
	#pragma omp parallel private(j)
	{
		#pragma omp for
		for (i=0;i<n;i++) {
			srand(i);
			arr[i]=(int*)malloc(n*sizeof(int));
			vec[i]=rand()%100;
			for (j=0;j<n;j++)
				arr[i][j]=rand()%100;
		}
	}
	#pragma omp parallel private(j)
	{
		#pragma omp for
		for(i=0;i<n;i++) {
			res[i]=0;
			for(j=0;j<n;j++)
				res[i]+=arr[i][j]*vec[j];
		}
	}
	printf("Matrix * Vector = Resultant Matrix\n");
	for(i=0;i<n;i++) {
		for(j=0;j<n;j++)
			printf("%3d ",arr[i][j]);
		if(i==n/2)
			printf("  *  %3d  = %6d\n",vec[i],res[i]);
```

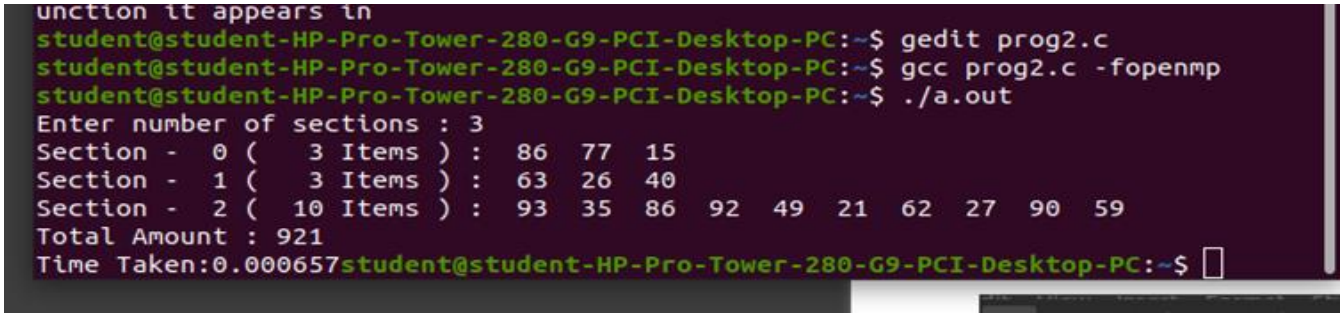
```
            else
                    printf("    %3d    %6d\n",vec[i],res[i]);
                    }
        et = clock();
        printf("Time Taken:%lf\n",(double)(et-st)/CLOCKS_PER_SEC);

}
```



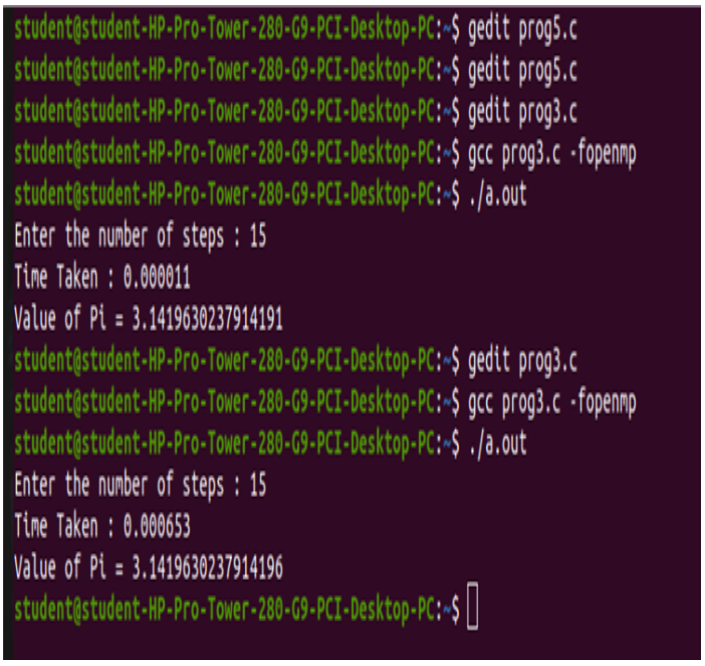
PROGRAM 2:

```
#include<stdio.h>
#include<stdlib.h>
#include<omp.h>
#include<time.h>
int main() {
   int r,i,ans=0;
   printf("Enter number of sections : ");
   scanf("%d",&r);
   time_t st, et;
   st=clock();
   int **arr=(int**)malloc(r*sizeof(int*));
   int *size=(int*)malloc(r*sizeof(int));
   omp_set_num_threads(r);
        #pragma omp parallel
   {
      #pragma omp for
      for (i=0;i<r;i++) {
         srand(i);
         int j,sum=0;
         size[i]=rand()%20;
                  arr[i]=(int*)malloc(size[i]*sizeof(int));
                  for (j=0;j<size[i];j++) {
                        arr[i][j]=rand()%100;
            sum+=arr[i][j];
         }
         #pragma omp critical
            ans+=sum;
            }
```

```
    }
    for(i=0;i<r;i++) {
        printf("Section - %2d ( %3d Items ) : ",i,size[i]);
        for(int j=0;j<size[i];j++)
            printf("%3d ",arr[i][j]);
        printf("\n");
    }
    printf("Total Amount : %d",ans);
    et=clock();
    printf("\nTime Taken:%lf",(double)(et-st)/CLOCKS_PER_SEC);
}
```



PROGRAM 3:




```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<omp.h>
void main() {
    int num,i;
    printf("Enter the number of steps : ");
    scanf("%d",&num);
    time_t st,et;
    st=clock();
    double step=1.0/(double)num,pi=0.0;
    omp_set_num_threads(num);
    #pragma omp parallel for reduction(+:pi)
    for(i=0;i<num;i++) {
        double x=(i+0.5)*step;
        double local_pi=(4.0*step)/(1+x*x);
        pi+=local_pi;
    }
    et=clock();
    printf("Time Taken : %lf\n",(double)((double)(et-st)/CLOCKS_PER_SEC));
    printf("Value of Pi = %.16lf\n",pi);
}
```

PROGRAM 4:

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
```

```
#include<omp.h>

void main() {
	int n, i;
	printf("Number of terms : ");
	scanf("%d",&n);
	int* a = (int*)malloc(n * sizeof(int));
	a[0] = 0;
	a[1] = 1;
	time_t st, et;
	st = clock();
	omp_set_num_threads(2);
	#pragma omp parallel
	{
		#pragma omp single
		{
			printf("id of thread involved in the computation of fibonacci numbers
= %d\n", omp_get_thread_num());
			for (i = 2; i < n; i++)
				a[i] = a[i - 2] + a[i - 1];
		}
		#pragma omp single
		{
			printf("id of thread involved in the displaying of fibonacci numbers = %d\n",
omp_get_thread_num());
			printf("Fibonacci numbers : ");
			for (i = 0; i < n; i++)
				printf("%d ", a[i]);
			printf("\n");
		}
	}
	et = clock();
	printf("Time Taken : %lfms\n", ((double)(et - st)*1000 / CLOCKS_PER_SEC));

}
\
```

```
Time Taken : 0.012000ms
student@student-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ gedit prog4.c
student@student-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ gcc prog4.c -fopenmp
student@student-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ ./a.out
Number of terms : 16
id of thread involved in the computation of fibonacci numbers = 0
id of thread involved in the displaying of fibonacci numbers = 0
Fibonacci numbers : 0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610
Time Taken : 0.057000ms
student@student-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ gedit prog4.c
student@student-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ gcc prog4.c -fopenmp
student@student-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ ./a.out
Number of terms : 16
id of thread involved in the computation of fibonacci numbers = 0
id of thread involved in the displaying of fibonacci numbers = 1
Fibonacci numbers : 0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610
Time Taken : 0.249000ms
student@student-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$
```

PROGRAM 5:

```
#include<stdio.h>
#include<stdlib.h>
```

```c
#include<time.h>
#include<omp.h>
void main() {
	int n, i;
	time_t st, et;
	st = clock();
	printf("Enter the number of students : ");
	scanf("%d", &n);
	double* arr = (double*)malloc(n * sizeof(double));
	double arr_max = 0;
	#pragma omp parallel for
	for (i = 0; i < n; i++) {
		srand(i);
		arr[i] = (double)(rand() % 10000)/10 ;
	}
	printf("CGPA of students : ");
	for (i = 0; i < n; i++)
		printf("%.2lf ", arr[i]);
	printf("\n");
	#pragma omp parallel for
	for (i = 0; i < n; i++) {
		#pragma omp critical
		if (arr_max < arr[i])
			arr_max = arr[i];
	}
	et = clock();
	printf("Student with highest CGPA = %.2lf\n", arr_max);
	printf("Time Taken : %.2lfms\n", ((double)(et - st) * 1000 / CLOCKS_PER_SEC));
}
```

```
collect2: error: ld returned 1 exit status
student@student-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ gedit prog5.c
student@student-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ gcc prog5.c -fopenmp
student@student-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ ./a.out
Enter the number of students : 5
CGPA of students : 938.30 529.00 671.90 474.60 830.10
Student with highest CGPA = 938.30
Time Taken : 2.25ms
student@student-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ gedit prog5.c
student@student-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ gcc prog5.c -fopenmp
student@student-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ ./a.out
Enter the number of students : 5
CGPA of students : 938.30 938.30 529.00 474.60 830.10
Student with highest CGPA = 938.30
Time Taken : 0.27ms
student@student-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$
```