CPU bound and a I/O bound programs

Aim: To execute CPU bound and I/O bound C program to observe the CPU performance.

CPU BOUND PROGRAM

Algorithm:

Step1: Start the program
Step2: Declare the variables start, end with datatype clock_t and variable runtime with double datatype.
Step3: Assign the clock() function to variable start.
Step4: Declare variables i, num=1 and primes=0
Step5: Repeat until num<=1000
    5.1: Initialize i=2
    5.2: Repeat until i<=num
        5.2.1 check num % i =0 then break
        5.2.2 : Increment i by 1.
    5.3: check num==i , increment primes by 1
    5.4: Clear console using system("clear")command
    5.5: Display the number of primes calculated
Step 6: Assign the clock() function to variable end
Step 7: Calculate runtime = (end-start)/(double)CLOCKS_PER_SEC
Step 8: Display runtime
Step 9: Stop the program

CPU BOUND C PROGRAM:

```
#include<stdio.h>
#include<time.h>
void main()
{
   clock_t start,end;
   double runTime;
   start =clock();
   int i, num=1, primes=0;
   int a;
   while(num <=1000)
   {
      i=2;
      while(i<=num)
      {
      if(num%i==0)
      break;
      i++;

   }
```

```
    if(i==num)
    primes++;
    system("clear");
    printf("%d primes number \n",primes);
    num++;
}
end=clock();
runTime =(end-start)/(double) CLOCKS_PER_SEC;
printf("This machine calculated all %d prime numbers under 100 in %g seconds
\n",primes,runTime);}
```

I/O BOUND PROGRAM

Algorithm:

Step1: Start the program
Step2: Declare the variables start, end with datatype clock_t and variable runtime with double datatype.
Step3: Assign the clock() function to variable start.
Step4: Declare variables i, a, num=1 and primes=0
Step5: Read a integer value for the variable 'a' from user
Step6: Repeat until num<= a
    5.1: Initialize i=2
    5.2: Repeat until i<=num
        5.2.1 check num % i =0 then break
        5.2.2 : Increment i by 1.
    5.3: check num==i , increment primes by 1
    5.4: Clear console using system("clear")command
    5.5: Display the number of primes calculated
Step 7: Assign the clock() function to variable end
Step 8: Calculate runtime = (end-start)/(double)CLOCKS_PER_SEC
Step 9: Display runtime
Step 10: Stop the program

I/O BOUND C PROGRAM

```
#include<stdio.h>
#include<time.h>
void main()
{
    clock_t start,end;
    double runTime;
    start =clock();
    int i, num=1, primes=0;
    int a;
    printf("Enter range:");
    scanf("%d",&a);
```

```
   while(num <=a)
   {
      i=2;
      while(i<=num)
          {
                    if(num%i==0)
                  break;
                  i++;
          }
   if(i==num)
   primes++;
   system("clear");
   printf("%d primes number \n",primes);
   num++;
  }
end=clock();
runTime =(end-start)/(double) CLOCKS_PER_SEC;
printf("This machine calculated all %d prime numbers under 1000  in %g seconds
\n",primes,runTime);}
```

Result: Thus the CPU bound and I/O bound programs executed successfully and the performance difference is observed.