Name:Ravindra Rinwa
Reg.No:20233230
section:C

Assignment-1

Q.1 Practice the following Linux commands: man, ls, cd, pwd, mkdir, rmdir, cat, less, head, tail, cp, mv, rm, touch, echo, ping, chmod, chown, exit, grep, diff, ps, top, kill, sudo, shutdown, vim, history, whoami, whatis, wc, more, cal, logout, I/O redirection commands (piping), etc.
Ans:

Name - Ravindra Rinwa

Reg. - 20233630

Q.1 Practice linux commands :-

1. man: Displays the manual page for a command (e.g man ls).

2. ls: Lists the files and directories in the current directory.

3. cd: Changes the current directory.

4. pwd: Prints the current working dir.

5. mkdir: creates a new directory.

6. rmdir: removes an empty directory.

7. cat : Display the content of file.

8. less : Opens a file for reading one page at a time.

9. head: Displays the first few lines of a file

10. tail: Displays the last few lines of a file.

11. cp: Copies files or directories

12. mv: Moves or renames file or directories

13. rm:- Removes the files or directories

14. touch : Creates an empty file or updates the time stamp of an existing file.

15. echo: Prints text or variables to the screen.

16. grep: Searches for a pattern in a file or output

17. diff: Compares the contents to two files line by line.

18. wc: Counts words, lines, or characters in a file.

19. history : Display the history of commands run in the terminal.

20. ps: Display information about running process

21. top: provides a real-time view of system processes and resource usage.

22. kill: Terminates a process by its PID

23 sudo : Runs commands as a superuser or elevated privileges.

24. Shutdown : Powers off or restarts the system.

25. whoami : Displays the current user

Ravinhu Rihwn
2123630

26. whatis : Provides a brief description of a command

27. logout : logs out of the current user session

28 vim : Opens the vim text editor

29 more : views file contents one screen at a time

30. cal : Displays a calender for the current or specified month/year

31. ping : Sends ICMP packets to test network connectivity to a host .

32. chmod : Changes file or directory permission

33. chown : Changes the owner of a file or directory

34. piping (|) : passes the output of one command as input to another.

35 Redirection (>, >>, <) Redirect the output to a file or input from a file

- > : overwrites a file
- >> : Appends to a file
- < : Redirects input

2. Use the commands to show the following information of your system:
(i) CPU information
(ii) Memory information

Ans:

ow CPU information and memory information.

(i) CPU information

1. cat /proc/cpuinfo

   - Display detailed information about CPU(s) such as model name, clock speed, and cores.

2. ls cpu

   - Provide a summary of CPU Architecture, cores, and thr

3. top or htop

   - Display real-time information about CPU usage.

(ii) memory information

1. cat /proc/meminfo

   - Shows detailed memory statics, including total and available memory.

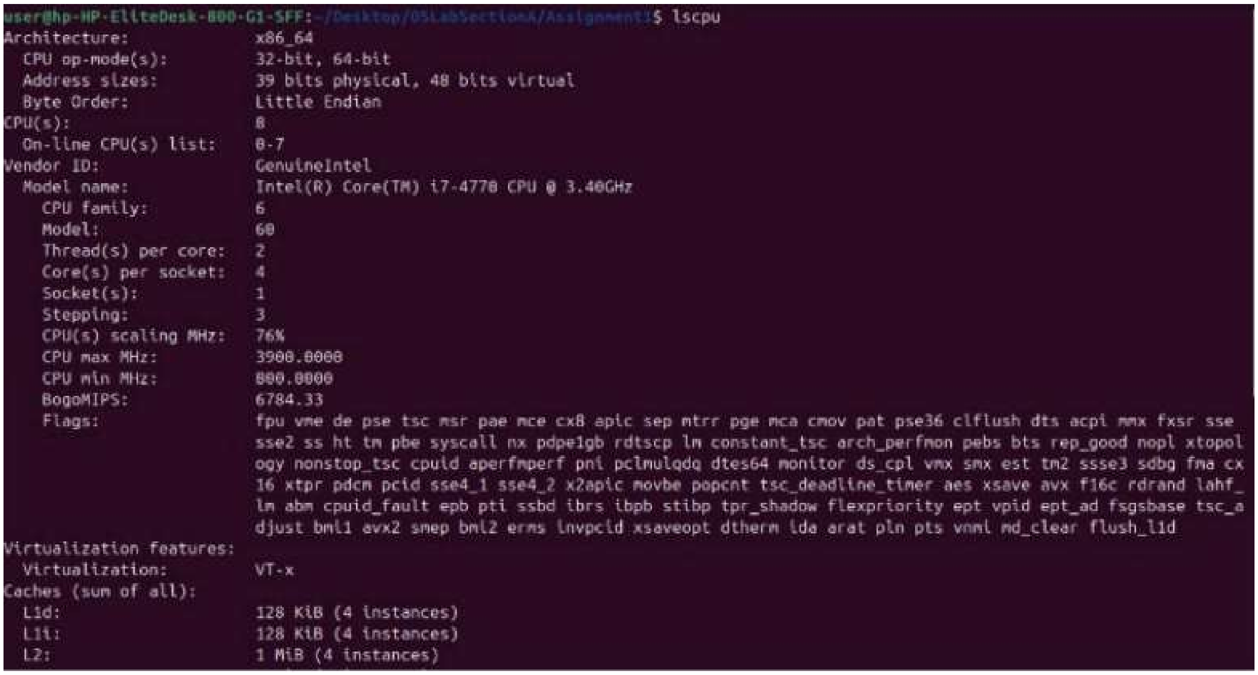
2. free -h

   - Displays memory usage in a human-readable format

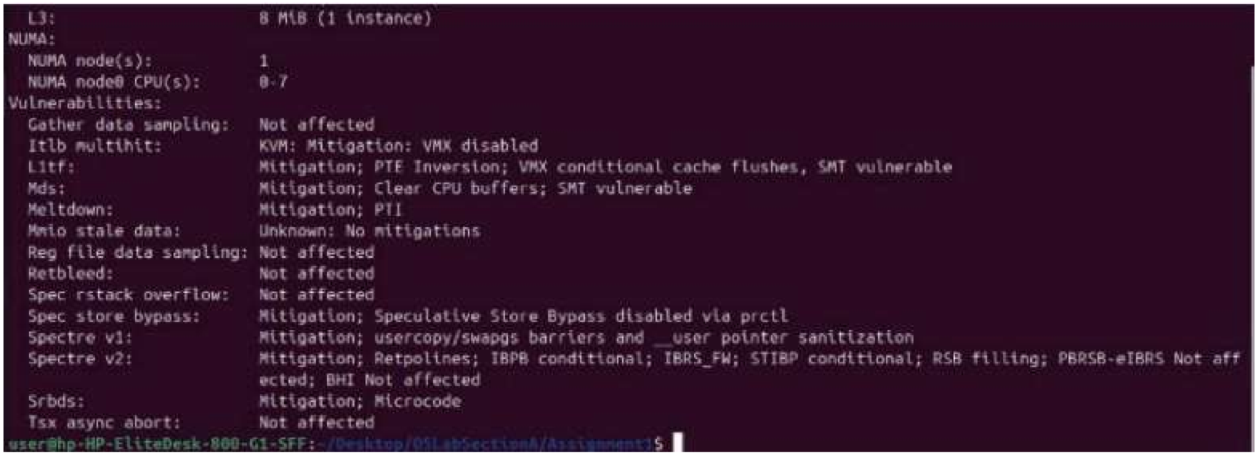
3. top & htop

   - Display real-time memory usage alongside and process details.

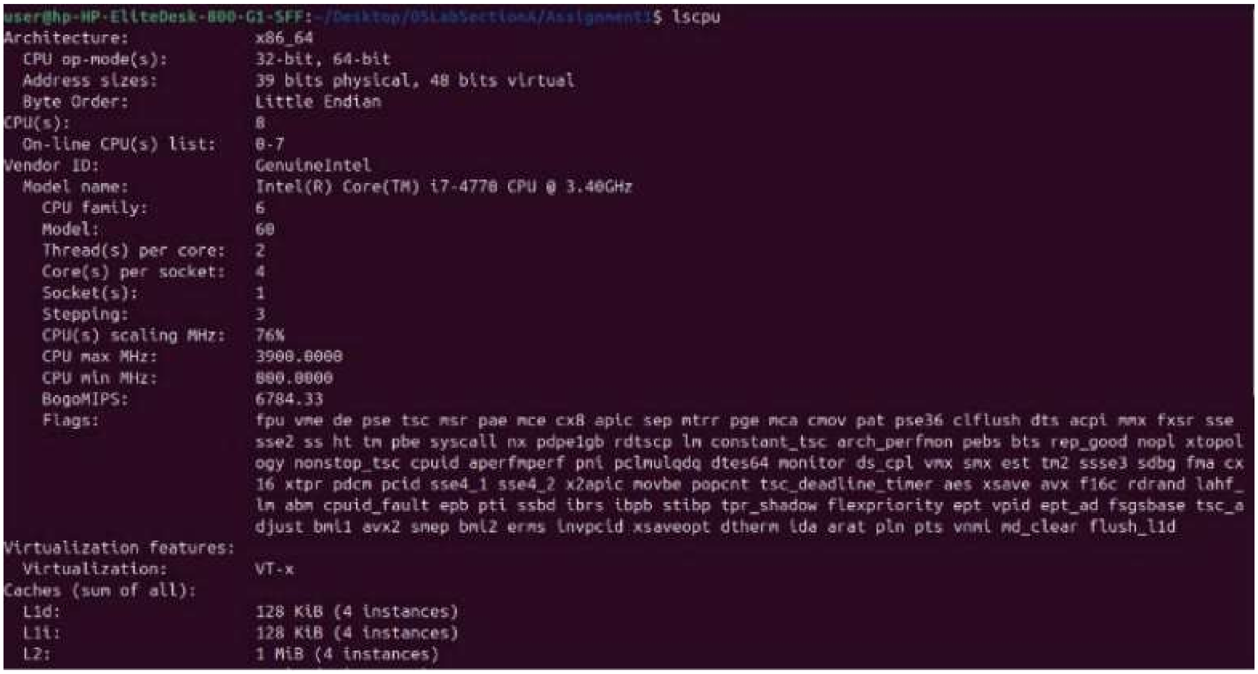
Output:



```
user@hp-HP-EliteDesk-800-G1-SFF:~/Desktop/OSLabSectionA/Assignment1$ lscpu
Architecture:            x86_64
  CPU op-mode(s):        32-bit, 64-bit
  Address sizes:         39 bits physical, 48 bits virtual
  Byte Order:            Little Endian
CPU(s):                  8
  On-line CPU(s) list:   0-7
Vendor ID:               GenuineIntel
  Model name:            Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz
    CPU family:          6
    Model:               60
    Thread(s) per core:  2
    Core(s) per socket:  4
    Socket(s):           1
    Stepping:            3
    CPU(s) scaling MHz:  76%
    CPU max MHz:         3900.0000
    CPU min MHz:         800.0000
    BogoMIPS:            6784.33
    Flags:               fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
                         sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtopol
                         ogy nonstop_tsc cpuid aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 sdbg fma cx
                         16 xtpr pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_
                         lm abm cpuid_fault epb pti ssbd ibrs ibpb stibp tpr_shadow flexpriority ept vpid ept_ad fsgsbase tsc_a
                         djust bmi1 avx2 smep bmi2 erms invpcid xsaveopt dtherm ida arat pln pts vnmi md_clear flush_l1d
Virtualization features:
  Virtualization:        VT-x
Caches (sum of all):
  L1d:                   128 KiB (4 instances)
  L1i:                   128 KiB (4 instances)
  L2:                    1 MiB (4 instances)
```



```
  L3:                    8 MiB (1 instance)
NUMA:
  NUMA node(s):          1
  NUMA node0 CPU(s):     0-7
Vulnerabilities:
  Gather data sampling:  Not affected
  Itlb multihit:         KVM: Mitigation: VMX disabled
  L1tf:                  Mitigation; PTE Inversion; VMX conditional cache flushes, SMT vulnerable
  Mds:                   Mitigation; Clear CPU buffers; SMT vulnerable
  Meltdown:              Mitigation; PTI
  Mmio stale data:       Unknown: No mitigations
  Reg file data sampling: Not affected
  Retbleed:              Not affected
  Spec rstack overflow:  Not affected
  Spec store bypass:     Mitigation; Speculative Store Bypass disabled via prctl
  Spectre v1:            Mitigation; usercopy/swapgs barriers and __user pointer sanitization
  Spectre v2:            Mitigation; Retpolines; IBPB conditional; IBRS_FW; STIBP conditional; RSB filling; PBRSB-eIBRS Not aff
                         ected; BHI Not affected
  Srbds:                 Mitigation; Microcode
  Tsx async abort:       Not affected
user@hp-HP-EliteDesk-800-G1-SFF:~/Desktop/OSLabSectionA/Assignment1$ 
```



```
user@hp-HP-EliteDesk-800-G1-SFF:~/Desktop/OSLabSectionA/Assignment3$ lscpu
Architecture:            x86_64
  CPU op-mode(s):        32-bit, 64-bit
  Address sizes:         39 bits physical, 48 bits virtual
  Byte Order:            Little Endian
CPU(s):                  8
  On-line CPU(s) list:   0-7
Vendor ID:               GenuineIntel
  Model name:            Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz
    CPU family:          6
    Model:               60
    Thread(s) per core:  2
    Core(s) per socket:  4
    Socket(s):           1
    Stepping:            3
    CPU(s) scaling MHz:  76%
    CPU max MHz:         3900.0000
    CPU min MHz:         800.0000
    BogoMIPS:            6784.33
    Flags:               fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
                         sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtopol
                         ogy nonstop_tsc cpuid aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 sdbg fma cx
                         16 xtpr pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_
                         lm abm cpuid_fault epb pti ssbd ibrs ibpb stibp tpr_shadow flexpriority ept vpid ept_ad fsgsbase tsc_a
                         djust bmi1 avx2 smep bmi2 erms invpcid xsaveopt dtherm ida arat pln pts vnmi md_clear flush_l1d
Virtualization features:
  Virtualization:        VT-x
Caches (sum of all):
  L1d:                   128 KiB (4 instances)
  L1i:                   128 KiB (4 instances)
  L2:                    1 MiB (4 instances)
```

3. Write C programs to simulate the following Linux commands:
(i) cd
(ii) ls
(iii) mkdir
 (iv) grep
Ans:

Ravindra Rinwa

20233630

3. Write c program to simulate follow. linux command
   (i) cd   (iii) mkdir
   (ii) ls   (iv) grep

ans:
```
#include <stdio.h>        // Simulate cd command
#include <unistd.h>

int main (int argc, char *argv[]) {

        if (argc != 2) {
          printf(" Usage : %s <directory\n", argv[0]);
          return1;
        }

        if ( chdir(argv[1]) == 0) {
             printf(" changed directory to %s \n", argv[1]);
        } else {
             perror ("chdir failed");

        return 0;
}
```

(ii) Simulate ls command

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

int main (int argc, char* argv[]) {
    const char *path = (argc > 1) ? argc[1] : ".";
    struct dirent *entry;
    DIR *dir = opendir (path);
if ( !dir) {
    perror ("opendir failed\n");
    return 1;
}

while ( (entry = readdir(dir) ) != NULL) {
        Printf ( "%s\n", entry->d_name);
}

closedir ( dir)
return 0;
}
```

3) Simulate mkdir command:

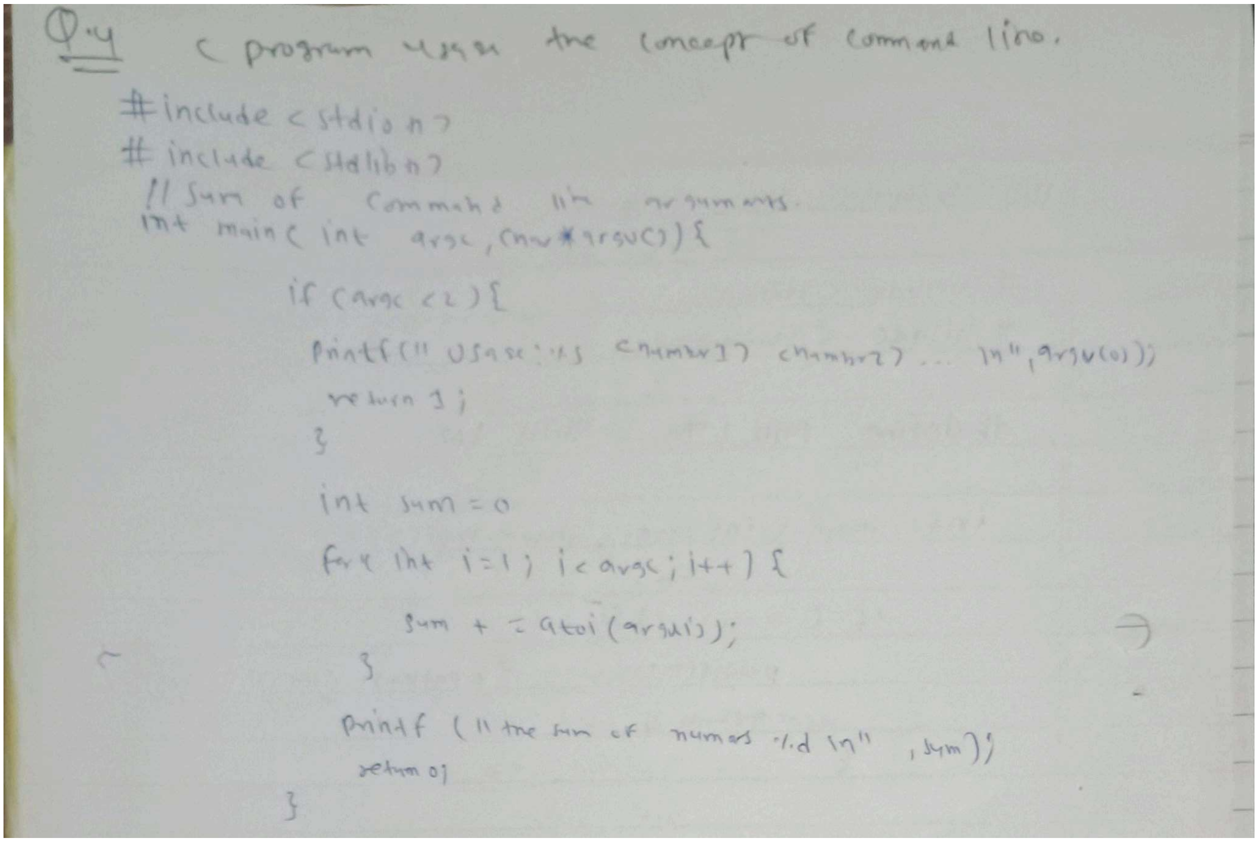
```
# include <stdio.h>
# include <sys/stat.h>
# include <sys/types.h>

int main ( int argc, char * argv[]) {
    if (argc != 2) {
        Printf( "Usage : %s directory\n", argv[0] );
        return1;
    }

    if ( mkdir(argv[1], 0755) == 0) {
        Printf(" Directory '%s' created successfully.\n", argv[1]);
    } else {
        perror( "mkdir failed");
```

Ravindu Pinna
20233030

(iv) Simulate grep command

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

#define MAX_LINE_LENGTH 1024

int main ( int argc, char *argv[]) {

    if ( argc!=3) {
        printf("Usage: %s <pattern> <file>\n", argv[0]);
        return 1;
    }

    Const char *Pattern = argv[1]
    Const char* filename = argv[2]
    FILE * file = fopen( filename, "r");

    if (!file) {
        perror ("fopen failed");
        return 1;
    }

    char line [MAX_LINE_LENGTH];
    while ( fgets ( line, Sizeof (line), file)) {
            if ( strstr( line, pattern)) {
                printf ("%s", line);
            }
    }
    fclose ( file);
    return 0;
}
```

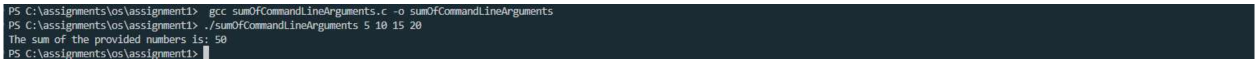
4. Write a C program which usage the concept of command line arguments.



Q.4 C program usage the concept of command line.

```
#include <stdio.h>
#include <stdlib.h>
// Sum of command line arguments.
int main( int argc, char *argv[]){

        if (argc <2){

         printf(" Usage: %s <number1> <number2> ... \n", argv[0]);

         return 1;

        }

        int sum = 0

        for( int i=1; i<argc; i++) {

              sum += atoi(argv[i]);

          }

         printf (" the sum of numbers %d \n" , sum);
          return 0;
      }
```

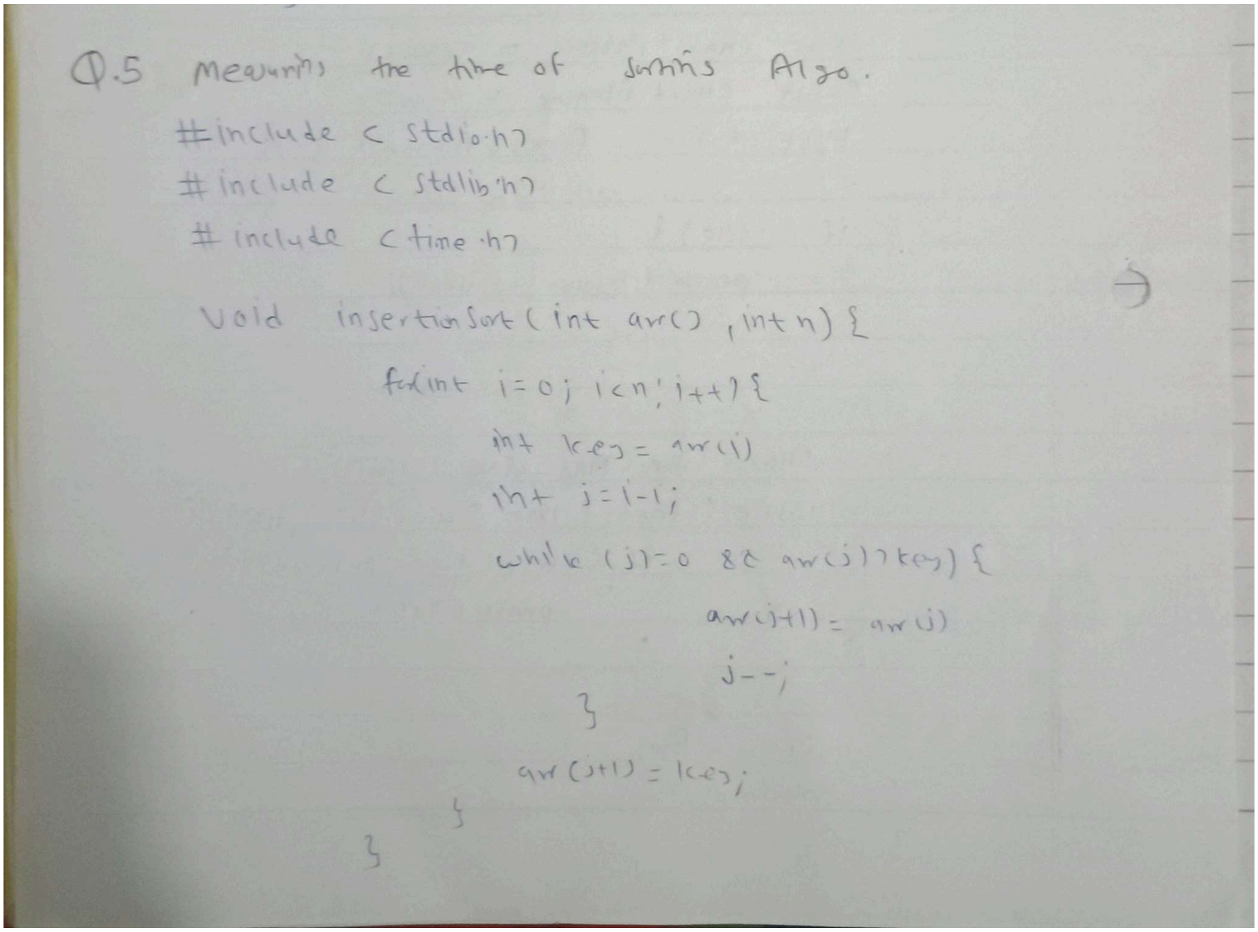
Output:



```
PS C:\assignments\os\assignment1>  gcc sumOfCommandLineArguments.c -o sumOfCommandLineArguments
PS C:\assignments\os\assignment1> ./sumOfCommandLineArguments 5 10 15 20
The sum of the provided numbers is: 50
PS C:\assignments\os\assignment1>
```

5. Write a C program to calculate the execution time taken by insertion sort, selection sort and bubble sort to sort the registration numbers of students of your class. You can use the appropriate data structures and functions from time.h header file.

Ans:

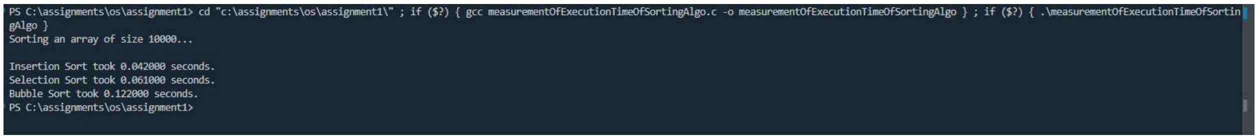


Q.5 Measuring the time of sorting Algo.

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

void  insertionSort(int arr[], int n) {
        for(int i=0; i<n; i++){
            int key = arr[i]
            int j=i-1;
            while (j>=0 && arr[j]>key) {
                    arr[j+1] = arr[j]
                    j--;
            }
            arr[j+1] = key;
        }
}
```

Ravindu Rimush
20233250

```
void  selectionSort (int arr(), int n){

        for( int i=0; i< n-1; i++){
            int  min Index = i

            for( int j = i+1; j<n; j++){
                  if(arr(j) <arr(minIndx)){
                        minIndx = j;
                  }
            }
            int  temp = arr(minIndex
            arr minIndx = arr(i)
              arr(i) = temp;
        }
   }

Void  bubble sort ( int  arr(), int n) {

        for( int  i= 0; i <n-1; i++){
            for(int j=0; j <n-i-1; j++){

                  if (arr(j) > arr(j+1)){
                        int  temp = arr(j);
                        arr(j) = arr(j+1);
                        arr(j+1) = temp;
                  }
            }
        }
  }
Void  measure Execution Time ( void (* Sort function) ( int [], int ), int arr[] , int n,
    const   char *  Sort Name ) {

        int *  tempArray  =  (int*) malloc ( n*  sizeof(int));

         for  (int   i=0; i<n; i++) {
                tempArray = arr[i];
            }

            clock_t   start  =  clock();
            sortfunction ( tempArray , n) ;
            Clock _t   end  = clock())

            double   timeTaken = (double )(end - start) / clocks_per_sec;
             printf (" %s took %.6f seconds .\n", SortName , timeTaken );

             free (temp Array);

       }

 int   main () {
        int  registrationNumber [] = { 2021034, 2021011, 2021036 --- 049 ,032, 028}
        int n =    size of  ( registrationNumbers) / size of ( registrationNumber [0]);

         Print (" original Array: \n");
         for( int i=0; i<n; i++) {
                Printf ( "%d ", registrationNumbers[i] );
          printf ("\n\n");

         measure ExecutionTime ( insertion Sort , registrationNumber , n, "insertion sort"))
         measure ExecutionTime ( selectionSort , registrationNumber n, "selection sort" );
         measure Execution Time ( bubble Sort , registrationNumber , n, "bubble sort" );
         return 0;
  }
```



```
PS C:\assignments\os\assignment1> cd "c:\assignments\os\assignment1\" ; if ($?) { gcc measurementOfExecutionTimeOfSortingAlgo.c -o measurementOfExecutionTimeOfSortingAlgo } ; if ($?) { .\measurementOfExecutionTimeOfSortin
gAlgo }
Sorting an array of size 10000...

Insertion Sort took 0.042000 seconds.
Selection Sort took 0.061000 seconds.
Bubble Sort took 0.122000 seconds.
PS C:\assignments\os\assignment1>
```