Lab Assignment-1

Indian Institute of Technology Roorkee Department of Computer Science and Engineering

CSN-361: Computer Networks Laboratory (Autumn 2019-2020)

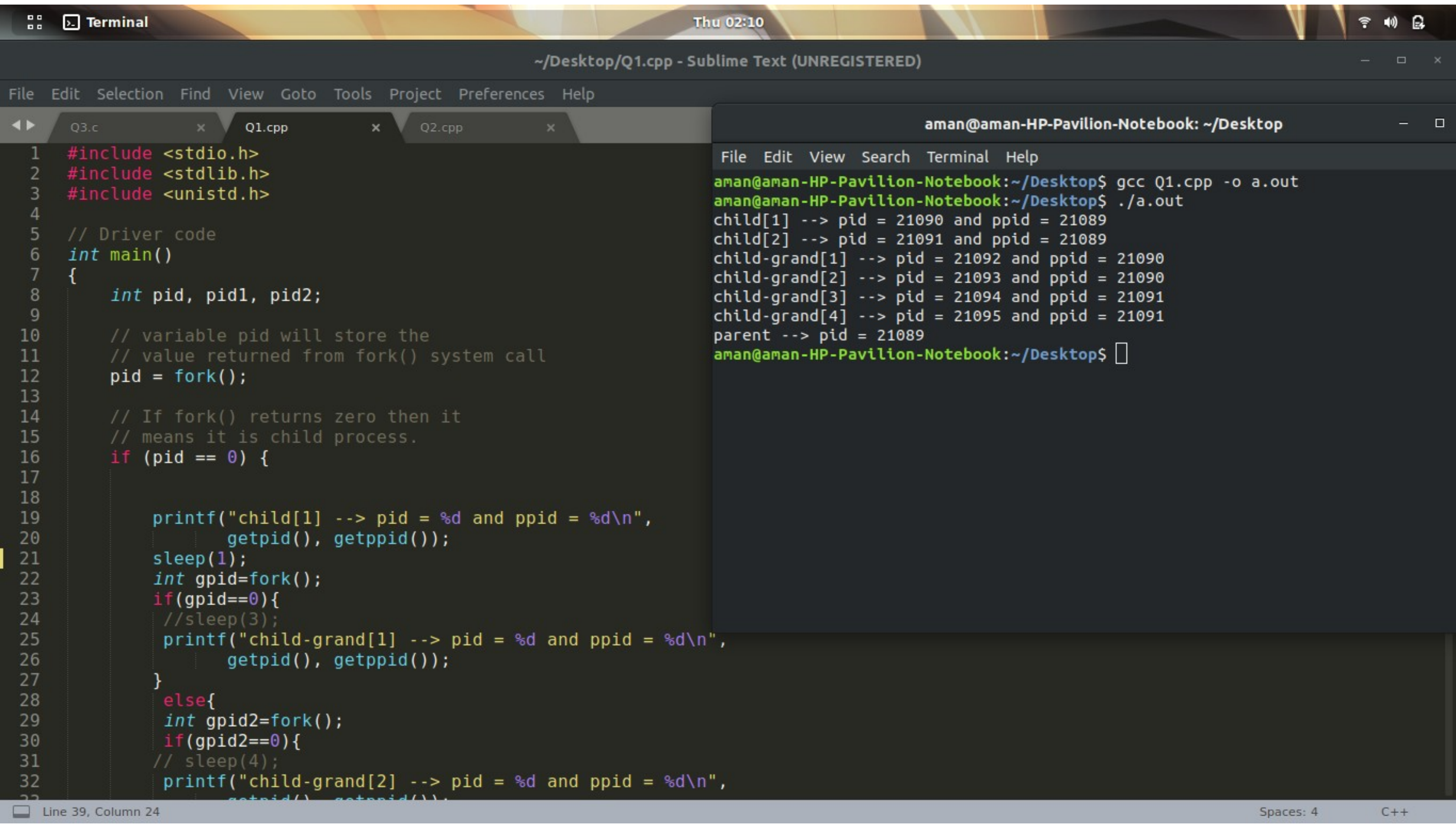
Aman jaiswal
Enrollment No:-17114008
B.tech CSE 3rd Year.

Problem Statement 1:

Write a C program in the UNIX system that creates two children and four grandchildren (two for each child). The program should then print the process-IDs of the two children, four grandchildren and the parent in this order.

- Data Structure and Functions used:-

1. fork():-fork system call use for creates a new process, which is called child process, which runs concurrently with parent process.
2. sleep(): causes the calling thread to sleep either until the number ofm real-time seconds specified in seconds have elapsed or until a signal arrives which is not ignored.
3. getpid(): gets process ID,
4. getppid(): gets parent process ID.
5. Variables to store process ID of process.



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

// Driver code
int main()
{
    int pid, pid1, pid2;

    // variable pid will store the
    // value returned from fork() system call
    pid = fork();

    // If fork() returns zero then it
    // means it is child process.
    if (pid == 0) {


        printf("child[1] --> pid = %d and ppid = %d\n",
               getpid(), getppid());
        sleep(1);
        int gpid=fork();
        if(gpid==0){
         //sleep(3);
         printf("child-grand[1] --> pid = %d and ppid = %d\n",
               getpid(), getppid());
        }
         else{
         int gpid2=fork();
         if(gpid2==0){
        // sleep(4);
         printf("child-grand[2] --> pid = %d and ppid = %d\n",
```

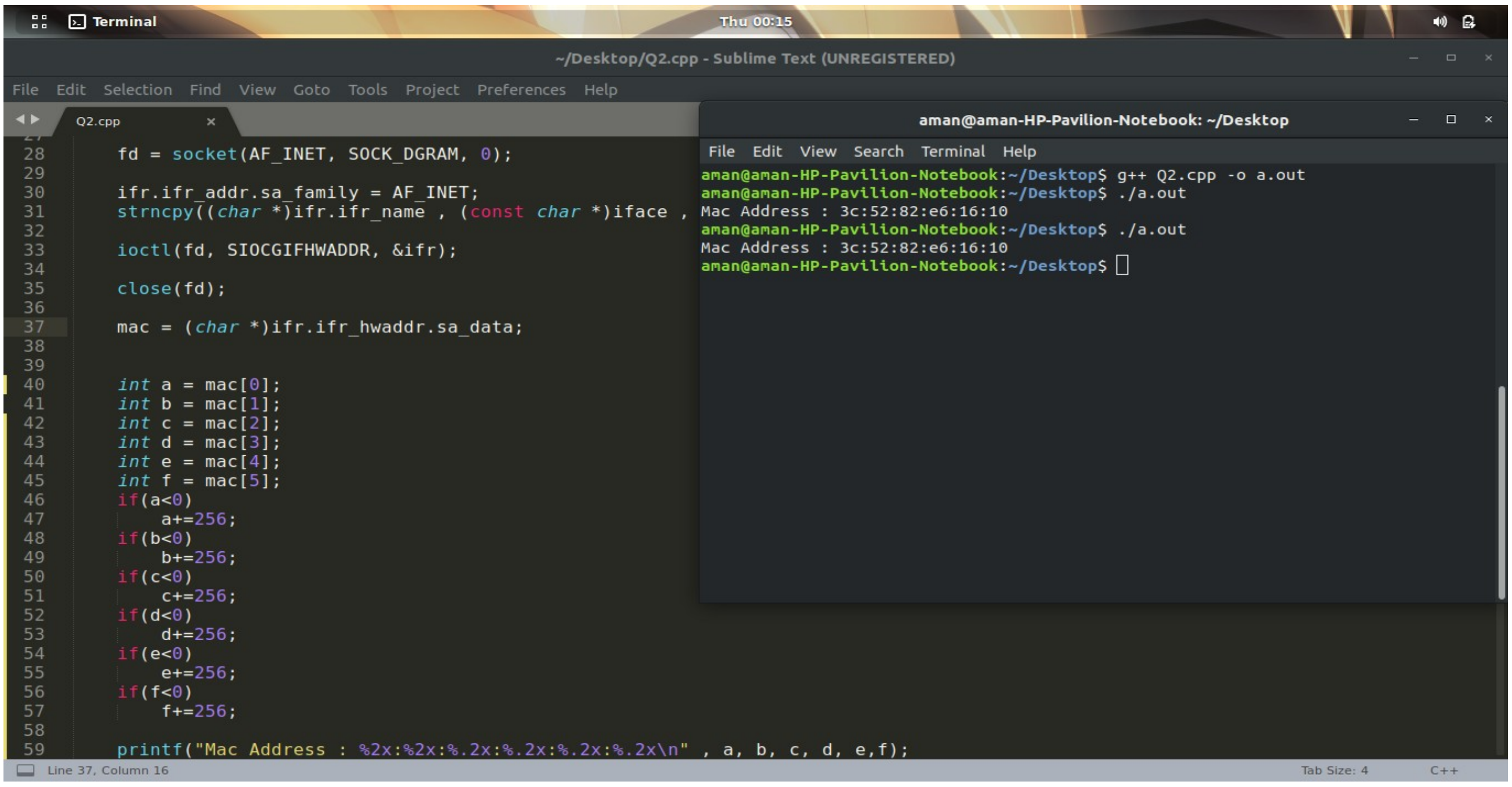
```
aman@aman-HP-Pavilion-Notebook:~/Desktop$ gcc Q1.cpp -o a.out
aman@aman-HP-Pavilion-Notebook:~/Desktop$ ./a.out
child[1] --> pid = 21090 and ppid = 21089
child[2] --> pid = 21091 and ppid = 21089
child-grand[1] --> pid = 21092 and ppid = 21090
child-grand[2] --> pid = 21093 and ppid = 21090
child-grand[3] --> pid = 21094 and ppid = 21091
child-grand[4] --> pid = 21095 and ppid = 21091
parent --> pid = 21089
aman@aman-HP-Pavilion-Notebook:~/Desktop$
```

Problem Statement 2:

Write a C++ program to print the MAC address of your computer.

- Data Structure and Functions used:-

1. The ioctl() Input-Output Control Command system call manipulates the underlying device parameter of special files.
2. socket() creates an endpoint for communication and returns a file descriptor that refers to that endpoint.
3. fd stores the socket file descriptor.
4. ifreq-Linux supports some standard ioctls to configure network devices.They can be used on any socket's file descriptor regardless of the family or type. Most of them pass an *ifreq* structure.
5. SIOCGIFHWADDR**:-**Get or set the hardware address of a device using ifr_hwaddr.



Terminal
Thu 00:15
~/Desktop/Q2.cpp - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
Q2.cpp
aman@aman-HP-Pavilion-Notebook: ~/Desktop
File Edit View Search Terminal Help
Line 37, Column 16
Tab Size: 4
C++

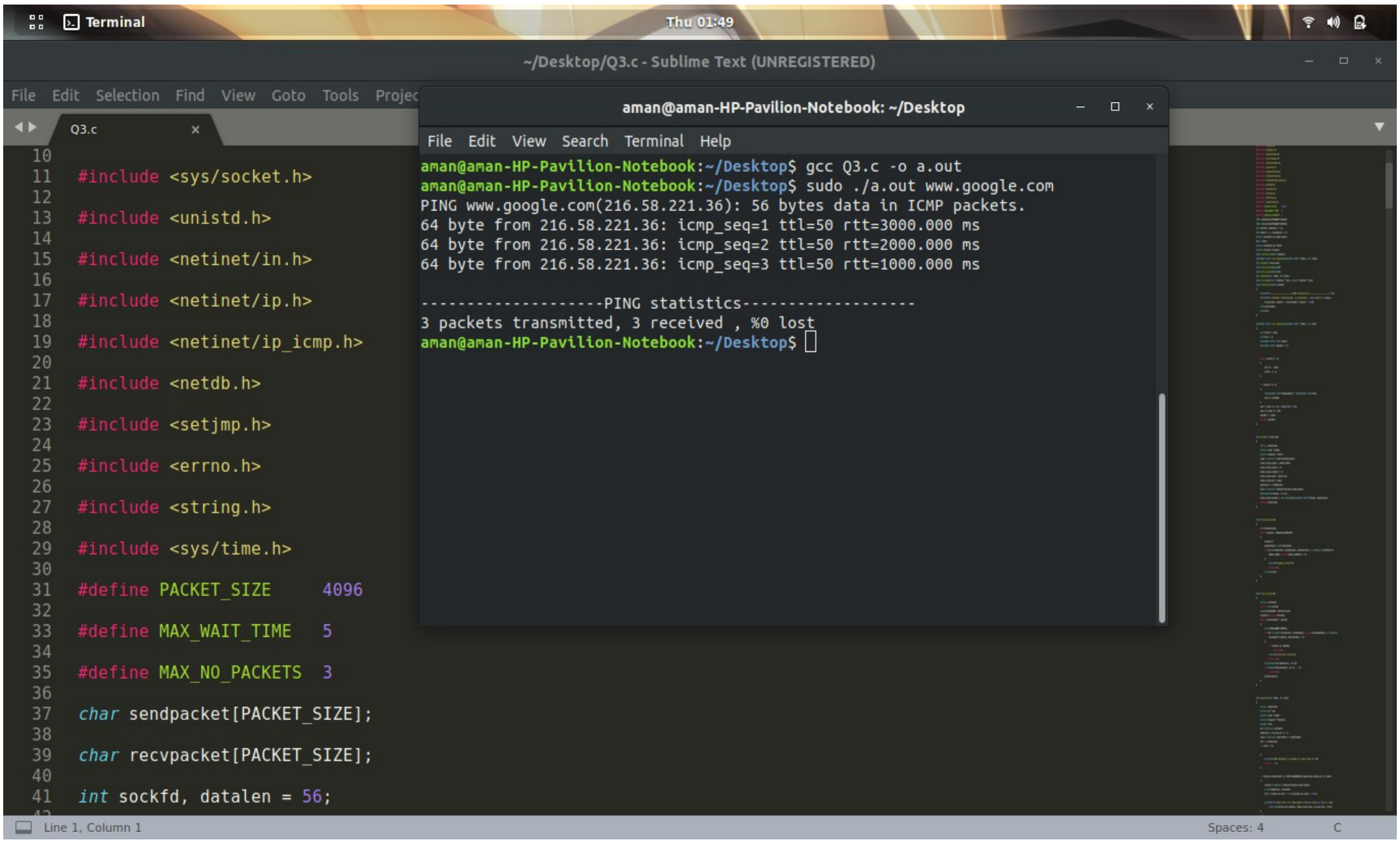
```
    fd = socket(AF_INET, SOCK_DGRAM, 0);

    ifr.ifr_addr.sa_family = AF_INET;
    strncpy((char *)ifr.ifr_name , (const char *)iface ,

    ioctl(fd, SIOCGIFHWADDR, &ifr);

    close(fd);

    mac = (char *)ifr.ifr_hwaddr.sa_data;


    int a = mac[0];
    int b = mac[1];
    int c = mac[2];
    int d = mac[3];
    int e = mac[4];
    int f = mac[5];
    if(a<0)
        a+=256;
    if(b<0)
        b+=256;
    if(c<0)
        c+=256;
    if(d<0)
        d+=256;
    if(e<0)
        e+=256;
    if(f<0)
        f+=256;

    printf("Mac Address : %2x:%2x:%.2x:%.2x:%.2x:%.2x\n" , a, b, c, d, e,f);
```

```
aman@aman-HP-Pavilion-Notebook:~/Desktop$ g++ Q2.cpp -o a.out
aman@aman-HP-Pavilion-Notebook:~/Desktop$ ./a.out
Mac Address : 3c:52:82:e6:16:10
aman@aman-HP-Pavilion-Notebook:~/Desktop$ ./a.out
Mac Address : 3c:52:82:e6:16:10
aman@aman-HP-Pavilion-Notebook:~/Desktop$
```

Problem Statement 3:

Write your own version of ping program in C language.

- Data Structure and Functions used:-
    1. gethostbyname: to get the IP address of the host.
    2. inet_addr() function converts the Internet host address cp from IPv4 numbers-and-dots notation into binary data in network byte order.
    3. getpid : system call of the process id.
    4. cal_cksum: code to calculate the checksum.
    5. recvfrom: calls are used to receive messages from a socket, and may be used to receive data on a socket whether or not it is connection-oriented.
    6. sendto: To send the data to the opened socket to the specified IP address.
    7. hostent: to store data about a specific host
    8. sock_addr_in: to specify a transport address and port for the AF_INET address family.
    9. timeval: checking interval for the socket.


Terminal
Thu 01:49
~/Desktop/Q3.c - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Projec
Q3.c
aman@aman-HP-Pavilion-Notebook: ~/Desktop
File Edit View Search Terminal Help
Line 1, Column 1
Spaces: 4
C

```c

#include <sys/socket.h>

#include <unistd.h>

#include <netinet/in.h>

#include <netinet/ip.h>

#include <netinet/ip_icmp.h>

#include <netdb.h>

#include <setjmp.h>

#include <errno.h>

#include <string.h>

#include <sys/time.h>

#define PACKET_SIZE     4096

#define MAX_WAIT_TIME   5

#define MAX_NO_PACKETS  3

char sendpacket[PACKET_SIZE];

char recvpacket[PACKET_SIZE];

int sockfd, datalen = 56;
```

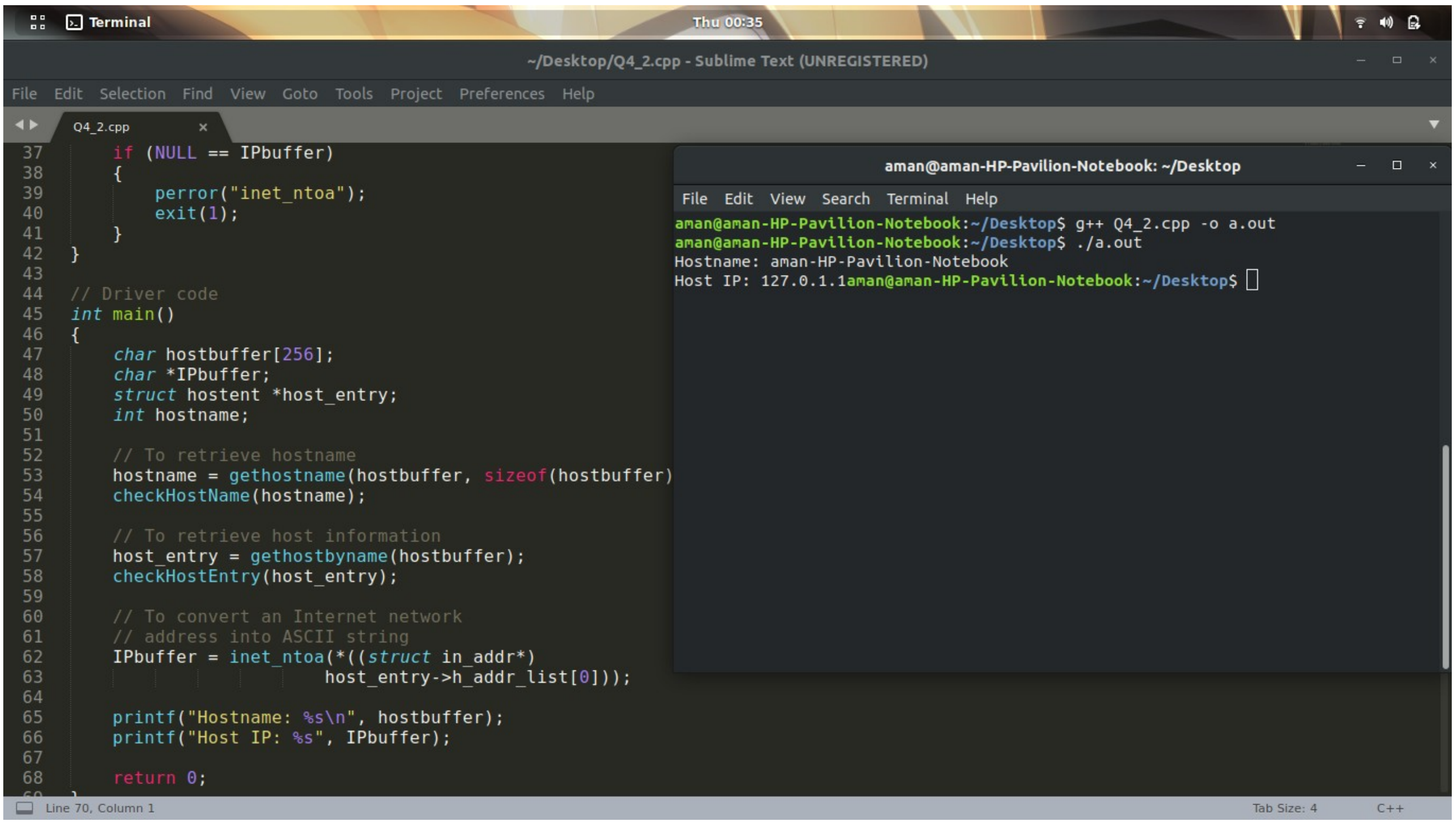
```
aman@aman-HP-Pavilion-Notebook:~/Desktop$ gcc Q3.c -o a.out
aman@aman-HP-Pavilion-Notebook:~/Desktop$ sudo ./a.out www.google.com
PING www.google.com(216.58.221.36): 56 bytes data in ICMP packets.
64 byte from 216.58.221.36: icmp_seq=1 ttl=50 rtt=3000.000 ms
64 byte from 216.58.221.36: icmp_seq=2 ttl=50 rtt=2000.000 ms
64 byte from 216.58.221.36: icmp_seq=3 ttl=50 rtt=1000.000 ms

-------------------PING statistics------------------
3 packets transmitted, 3 received , %0 lost
aman@aman-HP-Pavilion-Notebook:~/Desktop$ 
```

Problem Statement 4:

Write a C program to find the host name and the IP address of your computer.

- Data Structure and Functions used:-

  1. gethostname(): The gethostname function retrieves the standard host name for the local computer.
  2. gethostbyname(): The gethostbyname function retrieves host information corresponding to a host name from a host database.
  3. inet_ntoa(): The inet_ntoa function converts an (Ipv4) Internet network address into an ASCII string in Internet standard dotted-decimal format.
  4. Hostname:-Stores the host name.
  5. IPbuffer:- stores ip address.


Terminal
Thu 00:35
~/Desktop/Q4_2.cpp - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
Q4_2.cpp
aman@aman-HP-Pavilion-Notebook: ~/Desktop
File Edit View Search Terminal Help
Line 70, Column 1
Tab Size: 4
C++

```
    if (NULL == IPbuffer)
    {
        perror("inet_ntoa");
        exit(1);
    }
}

// Driver code
int main()
{
    char hostbuffer[256];
    char *IPbuffer;
    struct hostent *host_entry;
    int hostname;

    // To retrieve hostname
    hostname = gethostname(hostbuffer, sizeof(hostbuffer)
    checkHostName(hostname);

    // To retrieve host information
    host_entry = gethostbyname(hostbuffer);
    checkHostEntry(host_entry);

    // To convert an Internet network
    // address into ASCII string
    IPbuffer = inet_ntoa(*((struct in_addr*)
                        host_entry->h_addr_list[0]));

    printf("Hostname: %s\n", hostbuffer);
    printf("Host IP: %s", IPbuffer);

    return 0;
```

```
aman@aman-HP-Pavilion-Notebook:~/Desktop$ g++ Q4_2.cpp -o a.out
aman@aman-HP-Pavilion-Notebook:~/Desktop$ ./a.out
Hostname: aman-HP-Pavilion-Notebook
Host IP: 127.0.1.1aman@aman-HP-Pavilion-Notebook:~/Desktop$
```