CSN-361

Assignment 2

25.07.2019

VISHAL GARG
17114076

Problem Statement 2:

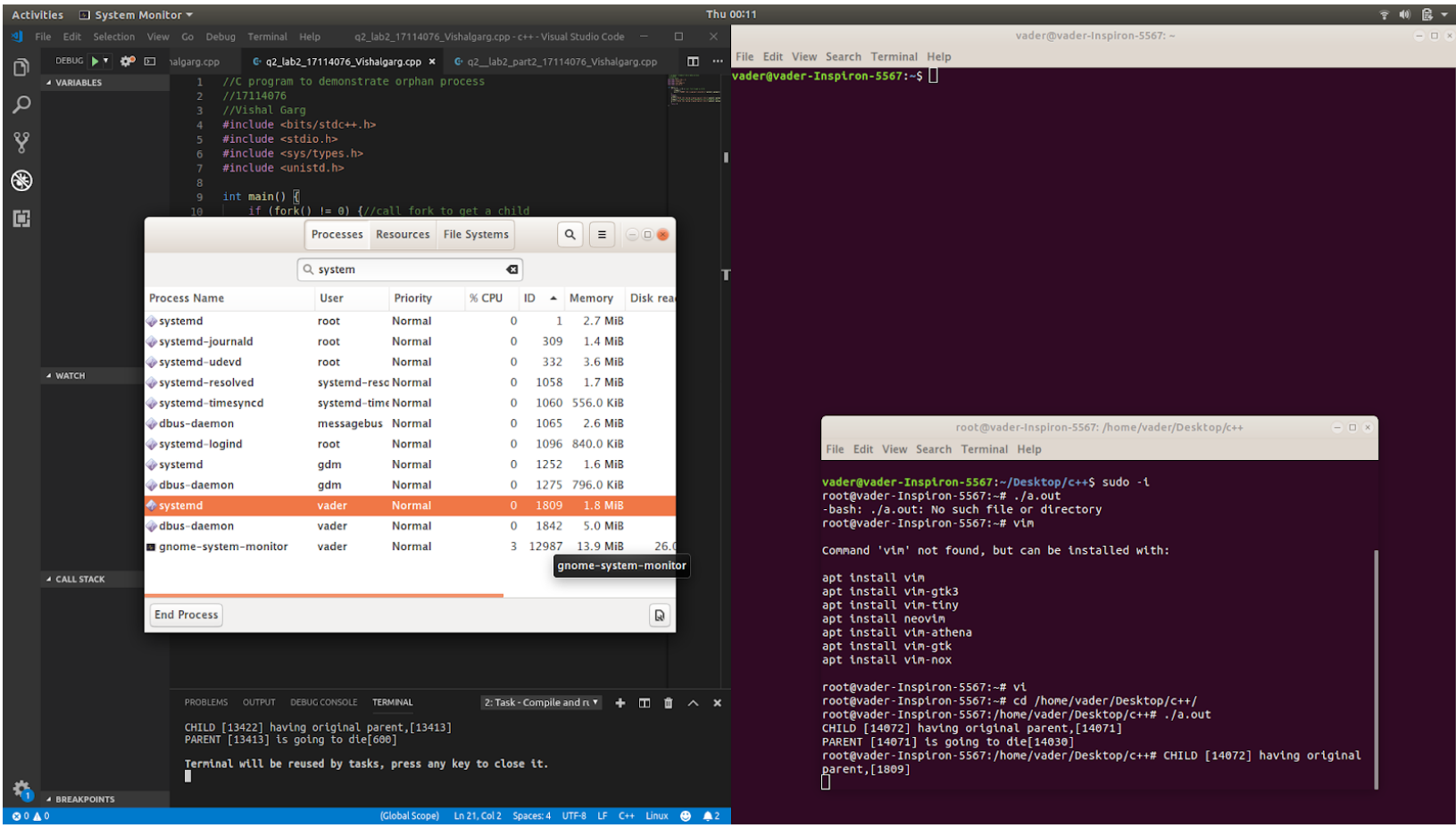
Write a C program to demonstrate both Zombie and Orphan process..

```c
//C program to demonstrate orphan process
//17114076
//Vishal Garg
#include <bits/stdc++.h>
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int main() {
    if (fork() != 0) {//call fork to get a child
       sleep(2);
       printf("PARENT [%d] is going to die[%d]\n",getpid(),getppid());//teminates before child rendering it orphan
    }
   else {
   sleep(1);
   printf("CHILD [%d] having original parent,[%d]\n",getpid(),getppid());
   sleep(3);//orphan process gets adopted by init/systemd process
   printf("CHILD [%d] having original parent,[%d]\n",getpid(),getppid());
    }
   return 0;
}
```

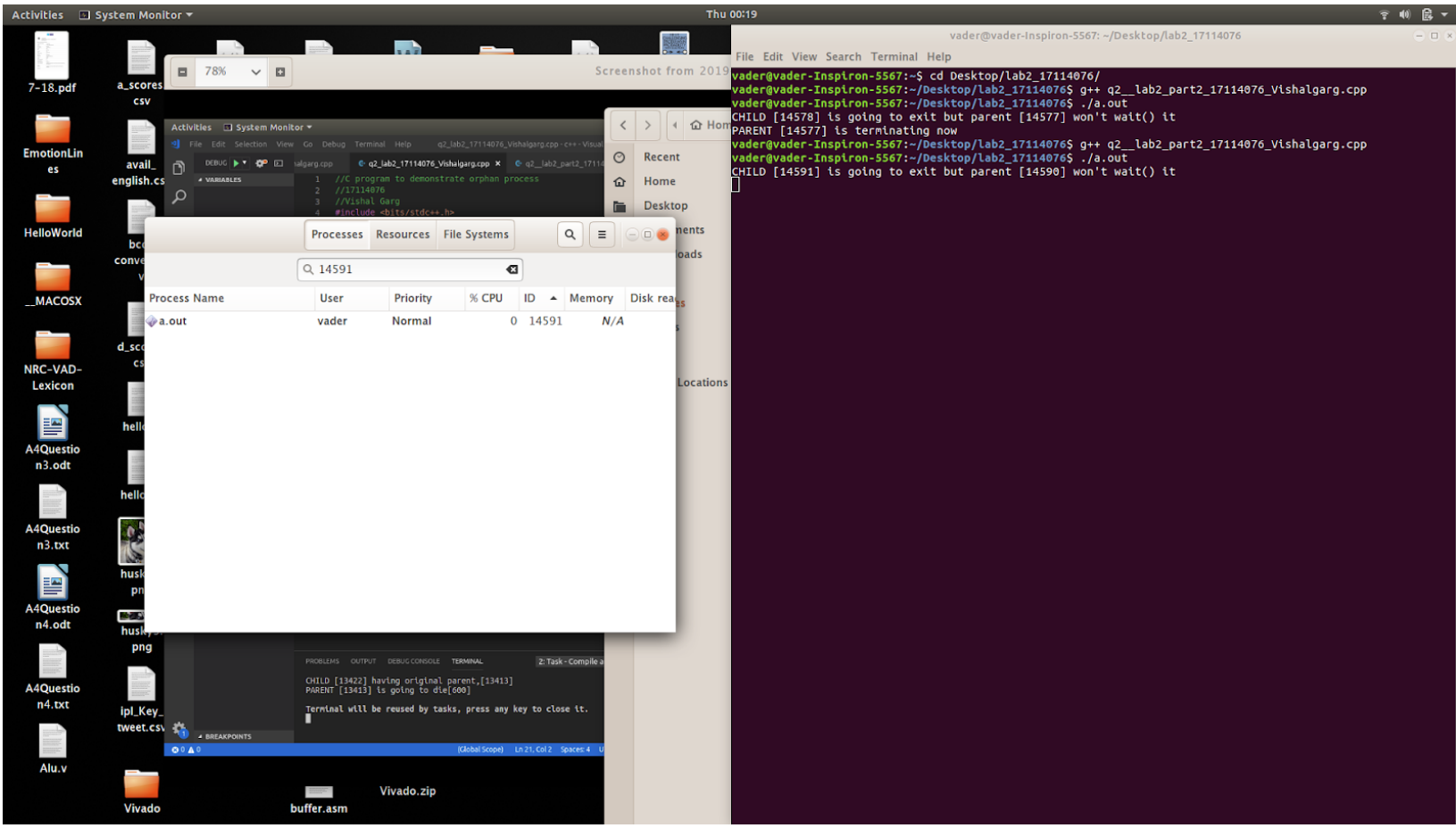
```c
//C program to demonstrate zombie process
//17114076
//Vishal Garg
#include <bits/stdc++.h>
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int main() {
        if (fork() != 0) {//fork to produce a child
       sleep(10);//parent is sleeping hence wont call wait() for child
       printf("PARENT [%d] is terminating now\n",getpid());
        } else {
       sleep(1);
       printf("CHILD [%d] is going to exit but parent [%d] won't wait()
it\n",getpid(),getppid());
       exit(0);//child has sent exit but signal hasnt be recieved by parent thus it stays
in process table
                }
    return 0;
}
```

Data Structure And algorithm

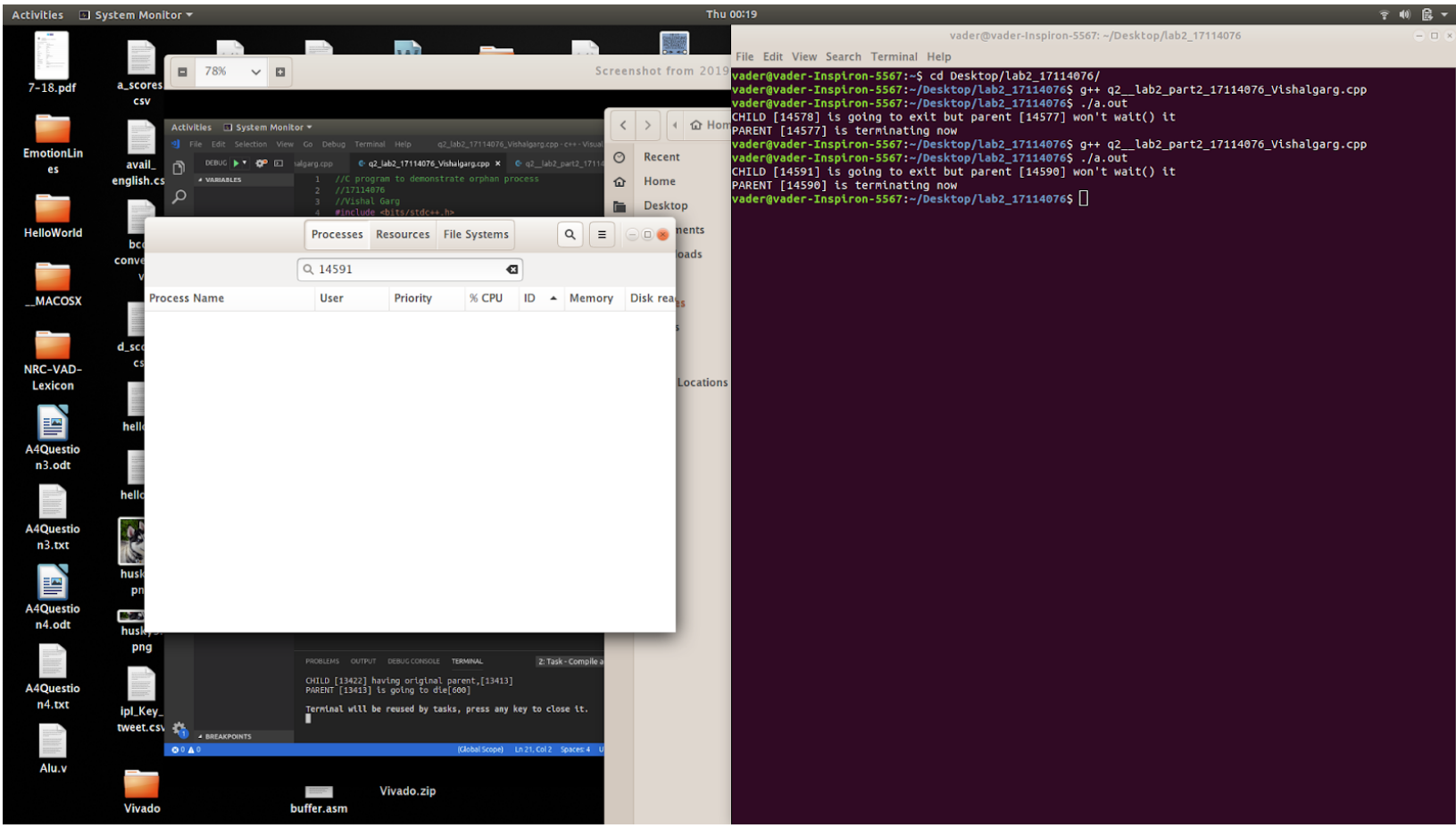
No complex algorithm has been used.



As you can see the child process has been adopted by systemd(init in case of linux-vader)



Child has completed, but is waiting on parent process to wait(), thus is present in process table as a zombie.


Activities
System Monitor
Thu 00:19
vader@vader-Inspiron-5567: ~/Desktop/lab2_17114076
File Edit View Search Terminal Help
vader@vader-Inspiron-5567:~$ cd Desktop/lab2_17114076/
vader@vader-Inspiron-5567:~/Desktop/lab2_17114076$ g++ q2__lab2_part2_17114076_Vishalgarg.cpp
vader@vader-Inspiron-5567:~/Desktop/lab2_17114076$ ./a.out
CHILD [14578] is going to exit but parent [14577] won't wait() it
PARENT [14577] is terminating now
vader@vader-Inspiron-5567:~/Desktop/lab2_17114076$ g++ q2__lab2_part2_17114076_Vishalgarg.cpp
vader@vader-Inspiron-5567:~/Desktop/lab2_17114076$ ./a.out
CHILD [14591] is going to exit but parent [14590] won't wait() it
PARENT [14590] is terminating now
vader@vader-Inspiron-5567:~/Desktop/lab2_17114076$
Processes
Resources
File Systems
14591
Process Name
User
Priority
% CPU
ID
Memory
Recent
Home
Desktop
Locations
7-18.pdf
EmotionLines
HelloWorld
__MACOSX
NRC-VAD-Lexicon
A4Question3.odt
A4Question3.txt
A4Question4.odt
A4Question4.txt
Alu.v
Vivado
buffer.asm
Vivado.zip

Problem Statement 1:

Write a socket program in C to connect two nodes on a network to communicate with each other, where one socket listens on a particular port at an IP, while other socket reaches out to the other to form a connection.

Node1-acting as a server

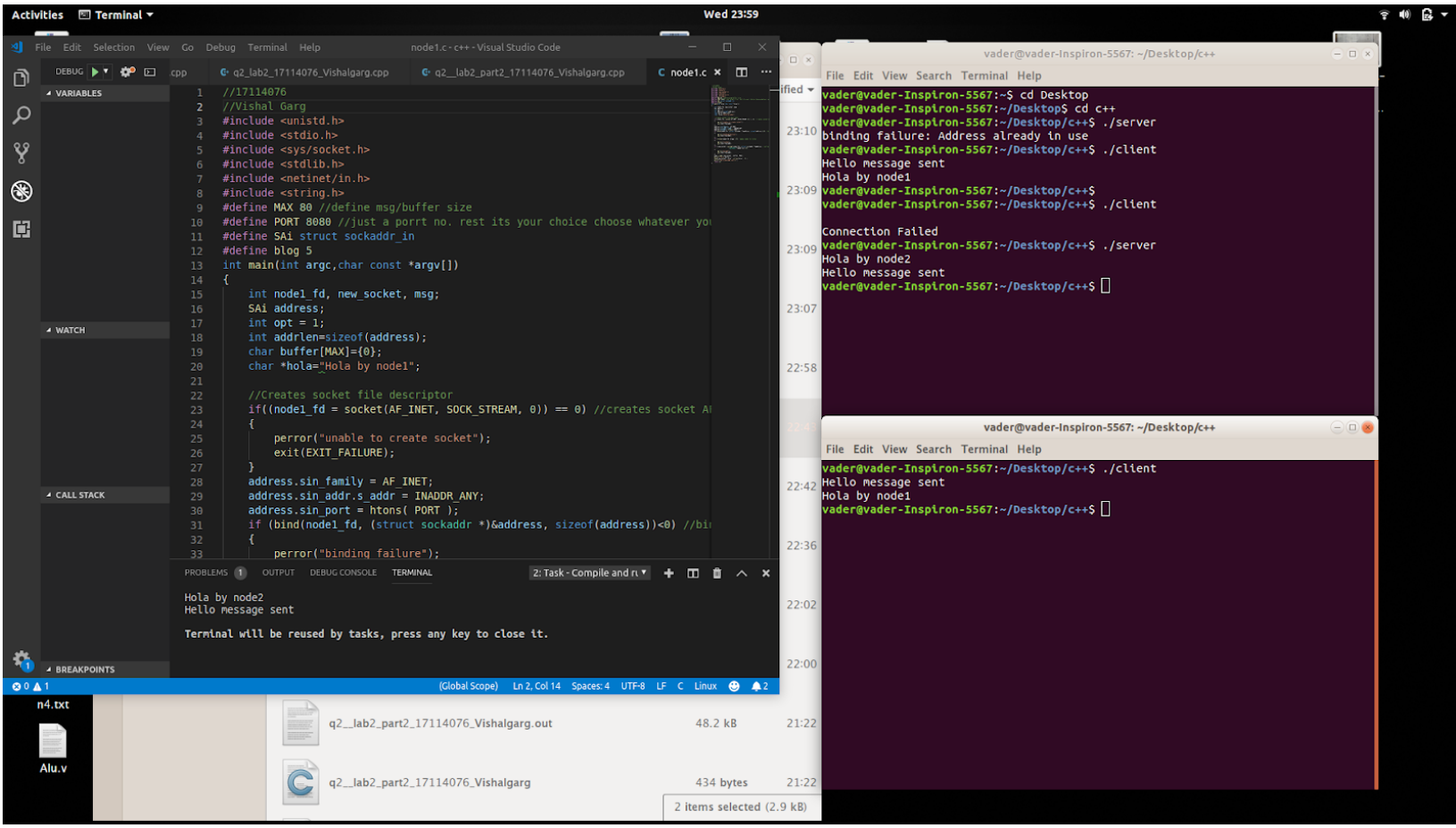
```
//17114076
//Vishal Garg
#include <unistd.h>
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <string.h>
#define MAX 80 //define msg/buffer size
#define PORT 8080 //just a porrt no. rest its your choice choose whatever you want to
#define SAi struct sockaddr_in
#define blog 5
int main(int argc,char const *argv[])
{
   int node1_fd, new_socket, msg;
   SAi address;
   int opt = 1;
   int addrlen=sizeof(address);
   char buffer[MAX]={0};
   char *hola="Hola by node1";

   //Creates socket file descriptor
   if((node1_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) //creates socket AF_NET signifies
IPV4,STREAM refers to TCP, and ) for IP
```

```
   {
       perror("unable to create socket");
       exit(EXIT_FAILURE);
   }
   address.sin_family = AF_INET;
   address.sin_addr.s_addr = INADDR_ANY;
   address.sin_port = htons( PORT );
   if (bind(node1_fd, (struct sockaddr *)&address, sizeof(address))<0) //binds to the address and
port number specified
   {
       perror("binding failure");
       exit(EXIT_FAILURE);
   }
   if (listen(node1_fd, blog) < 0) //makes node1 to listen
   {
       perror("listen");
       exit(EXIT_FAILURE);
   }
   if ((new_socket = accept(node1_fd, (struct sockaddr *)&address, //extracts first connection
req on the queue of pending connections and creates a new connected socket
                       (socklen_t*)&addrlen))<0)
   {
       perror("accept");
       exit(EXIT_FAILURE);
   }
   msg = read( new_socket , buffer, MAX);
   printf("%s\n",buffer );
   send(new_socket , hola , strlen(hola) , 0 );
```

```
   printf("Hello message sent\n");
   return 0;
}
```

```
//17114076
//Vishal Garg
#include <stdio.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <string.h>
#define MAX 80 //define msg/buffer size
#define PORT 8080 //just a porrt no. rest its your choice choose whatever you want to
#define SAi struct sockaddr_in

int main(int argc,char const *argv[])
{
   int node2_fd, msg;
   SAi s_address;
   char buffer[MAX]={0};
   char *hola="Hola by node2";

   //Creates socket file descriptor
   if((node2_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) //creates socket AF_NET signifies
IPV4,STREAM refers to TCP, and 0 for IP
```

```
    {
        perror("unable to create socket");
        return -1;
    }
    s_address.sin_family=AF_INET;
    s_address.sin_port = htons(PORT);

    if(inet_pton(AF_INET, "127.0.0.1", &s_address.sin_addr)<=0)
    {
        printf("\nInvalid address/ Address not supported \n");
        return -1;
    }
    if (connect(node2_fd, (struct sockaddr *)&s_address, sizeof(s_address)) < 0)
    {
        printf("\nConnection Failed \n");
        return -1;
    }
    send(node2_fd , hola , strlen(hola) , 0 );
    printf("Hello message sent\n");
    msg = read( node2_fd , buffer, MAX);
    printf("%s\n",buffer );
    return 0;
}
```



Data Structure And algorithm used:

Create TCP socket

using bind(), Bind the socket to server address

using listen(), put the server socket in a passive mode, where it waits for the client to approach the server to make a connection

using accept(), At this point, connection is established between client and server
they are now ready to transfer data.