PROGRAM:-

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <netdb.h>

#include <arpa/inet.h>

#define BUFFER\_SIZE 1024

void resolve\_hostname(const char \*hostname) {

struct addrinfo hints, \*res, \*p;

char ipstr[INET6\_ADDRSTRLEN];

// Prepare hints

memset(&hints, 0, sizeof hints);

hints.ai\_family = AF\_UNSPEC; // Use IPv4 or IPv6

hints.ai\_socktype = SOCK\_STREAM;

// Resolve the hostname

int status = getaddrinfo(hostname, NULL, &hints, &res);

if (status != 0) {

fprintf(stderr, "Error resolving hostname %s: %s\n", hostname, gai\_strerror(status));

return;

}

printf("Resolved IP addresses for %s:\n", hostname);

// Iterate over results and print IP addresses

for (p = res; p != NULL; p = p->ai\_next) {

void \*addr;

char \*ipver;

// Get pointer to address (IPv4 or IPv6)

if (p->ai\_family == AF\_INET) { // IPv4

struct sockaddr\_in \*ipv4 = (struct sockaddr\_in \*)p->ai\_addr;

addr = &(ipv4->sin\_addr);

ipver = "IPv4";

} else { // IPv6

struct sockaddr\_in6 \*ipv6 = (struct sockaddr\_in6 \*)p->ai\_addr;

addr = &(ipv6->sin6\_addr);

ipver = "IPv6";

}

// Convert IP to string and print

inet\_ntop(p->ai\_family, addr, ipstr, sizeof ipstr);

printf(" %s: %s\n", ipver, ipstr);

}

// Free memory allocated by getaddrinfo

freeaddrinfo(res);

}

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

if (argc != 2) {

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

return 1;

}

resolve\_hostname(argv[1]);

return 0;

}

OUTPUT:-

Resolved IP addresses for google.com:

IPv4: 142.250.190.14

IPv6: 2607:f8b0:4005:80a::200e