Assignment-4

1. Explain the connection procedure followed in client server communication

Soln:-

1. Client Initialization
2. Server Initialization
3. Connection Request
4. Server Acceptance
5. Data Exchange
6. Connection Termination

2. What is the use of bind() function in socket programming ?

Soln:-

bind() function is used to associate a socket with a specific network address, such as an IP address and port number. This function is essential for server applications, particulary when they need to listen for incomming connections on specific port.

3. What is Datagram Socket?

Soln:-

A Datagram socket is a type of socket used in network programming to facilitate communication using the User Datagram Protocol (UDP). Datagram socket offers connectionless unreliable communication method.

4. Write a server/client model socket program to exchange hello message between them.

Soln:-
server.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 12345
#define BUFFER_SIZE 1024

int main() {
    int server_fd, client_fd;
    struct sockaddr_in server_addr, client_addr;
    char buffer[BUFFER_SIZE] = {0};

    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }

    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = INADDR_ANY;
    server_addr.sin_port = htons(PORT);

    if (bind(server_fd, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {
        perror("bind failed");
        exit(EXIT_FAILURE);
    }

    if (listen(server_fd, 1) < 0) {
        perror("listen failed");
        exit(EXIT_FAILURE);
    }

    printf("Server is listening for connections...\n");

    int client_addr_len = sizeof(client_addr);
            if ((client_fd = accept(server_fd, (struct sockaddr *)&client_addr, (socklen_t
*)&client_addr_len)) < 0) {
        perror("accept failed");
        exit(EXIT_FAILURE);
    }

    printf("Connection established with client\n");
    char *hello_message = "Hello from server!";
    send(client_fd, hello_message, strlen(hello_message), 0);

    int valread;
    if ((valread = recv(client_fd, buffer, BUFFER_SIZE, 0)) == -1) {
        perror("recv failed");
        exit(EXIT_FAILURE);
    }

    printf("Received message from client: %s\n", buffer);
    close(client_fd);
    close(server_fd);

    return 0;
}
```

Client.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 12345
#define BUFFER_SIZE 1024

int main() {
    int client_fd;
    struct sockaddr_in server_addr;
    char buffer[BUFFER_SIZE] = {0};

    if ((client_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }

    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);

    if (inet_pton(AF_INET, "127.0.0.1", &server_addr.sin_addr) <= 0) {
        perror("Invalid address/ Address not supported");
        exit(EXIT_FAILURE);
    }
    if (connect(client_fd, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {
        perror("connect failed");
        exit(EXIT_FAILURE);
    }

    int valread;
    if ((valread = recv(client_fd, buffer, BUFFER_SIZE, 0)) == -1) {
        perror("recv failed");
        exit(EXIT_FAILURE);
    }
    printf("Received message from server: %s\n", buffer);
    char *hello_message = "Hello from client!";
    send(client_fd, hello_message, strlen(hello_message), 0);
    close(client_fd);

    return 0;
}
```

5. Write a TCP server-client program to check if the string is Palindrome.

Soln:

server.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 12345
#define BUFFER_SIZE 1024

int isPalindrome(char *str) {
    int length = strlen(str);
    for (int i = 0; i < length / 2; i++) {
        if (str[i] != str[length - i - 1]) {
            return 0; // Not a palindrome
        }
    }
    return 1; // Palindrome
}

int main() {
    int server_fd, client_fd;
    struct sockaddr_in server_addr, client_addr;
    char buffer[BUFFER_SIZE] = {0};
    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = INADDR_ANY;
    server_addr.sin_port = htons(PORT);
```

```
   if (bind(server_fd, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {
      perror("bind failed");
      exit(EXIT_FAILURE);
   }
   if (listen(server_fd, 1) < 0) {
      perror("listen failed");
      exit(EXIT_FAILURE);
   }

   printf("Server is listening for connections...\n");

   int client_addr_len = sizeof(client_addr);
   if ((client_fd = accept(server_fd, (struct sockaddr *)&client_addr, (socklen_t *)&client_addr_len)) < 0) {
      perror("accept failed");
      exit(EXIT_FAILURE);
   }

   printf("Connection established with client\n");

   int valread;
   if ((valread = recv(client_fd, buffer, BUFFER_SIZE, 0)) == -1) {
      perror("recv failed");
      exit(EXIT_FAILURE);
   }
   printf("Received string from client: %s\n", buffer);

   if (isPalindrome(buffer)) {
      char *palindrome_message = "Palindrome";
      send(client_fd, palindrome_message, strlen(palindrome_message), 0);
   } else {
      char *not_palindrome_message = "Not a Palindrome";
      send(client_fd, not_palindrome_message, strlen(not_palindrome_message), 0);
   }

   close(client_fd);
   close(server_fd);

   return 0;
}
```

Client.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 12345
#define BUFFER_SIZE 1024

int main() {
    int client_fd;
    struct sockaddr_in server_addr;
    char buffer[BUFFER_SIZE] = {0};

    if ((client_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    if (inet_pton(AF_INET, "127.0.0.1", &server_addr.sin_addr) <= 0) {
        perror("Invalid address/ Address not supported");
        exit(EXIT_FAILURE);
    }
    if (connect(client_fd, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {
        perror("connect failed");
        exit(EXIT_FAILURE);
    }
    printf("Enter a string: ");
    fgets(buffer, BUFFER_SIZE, stdin);
    buffer[strcspn(buffer, "\n")] = '\0'; // Remove newline character
    send(client_fd, buffer, strlen(buffer), 0);
    int valread;
    if ((valread = recv(client_fd, buffer, BUFFER_SIZE, 0)) == -1) {
        perror("recv failed");
        exit(EXIT_FAILURE);
    }
    printf("Server response: %s\n", buffer);
    close(client_fd);
    return 0;
}
```

6. Write an example to demonstrate UDP program?

Soln:

server.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 12345
#define BUFFER_SIZE 1024

int main() {
    int server_fd;
    struct sockaddr_in server_addr, client_addr;
    char buffer[BUFFER_SIZE] = {0};

    if ((server_fd = socket(AF_INET, SOCK_DGRAM, 0)) == 0) {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }

    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = INADDR_ANY;
    server_addr.sin_port = htons(PORT);

    if (bind(server_fd, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {
        perror("bind failed");
        exit(EXIT_FAILURE);
    }
    printf("Server is listening for messages...\n");

    int client_addr_len = sizeof(client_addr);
    int message_length;
    if ((message_length = recvfrom(server_fd, buffer, BUFFER_SIZE, 0, (struct sockaddr
*)&client_addr, (socklen_t *)&client_addr_len)) == -1) {
        perror("recvfrom failed");
        exit(EXIT_FAILURE);
    }

    printf("Received message from client: %s\n", buffer);

    if (sendto(server_fd, buffer, message_length, 0, (struct sockaddr *)&client_addr,
client_addr_len) == -1) {
        perror("sendto failed");
        exit(EXIT_FAILURE);
    }

    printf("Response sent to client\n");

    close(server_fd);

    return 0;
}
```

Client.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 12345
#define BUFFER_SIZE 1024

int main() {
    int client_fd;
    struct sockaddr_in server_addr;
    char buffer[BUFFER_SIZE] = {0};

    if ((client_fd = socket(AF_INET, SOCK_DGRAM, 0)) == 0) {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);

    if (inet_pton(AF_INET, "127.0.0.1", &server_addr.sin_addr) <= 0) {
        perror("Invalid address/ Address not supported");
        exit(EXIT_FAILURE);
    }

    printf("Enter a message: ");
    fgets(buffer, BUFFER_SIZE, stdin);
    buffer[strcspn(buffer, "\n")] = '\0'; // Remove newline character

    if (sendto(client_fd, buffer, strlen(buffer), 0, (struct sockaddr *)&server_addr,
sizeof(server_addr)) == -1) {
        perror("sendto failed");
        exit(EXIT_FAILURE);
    }

    printf("Message sent to server\n");

    int message_length;
    if ((message_length = recvfrom(client_fd, buffer, BUFFER_SIZE, 0, NULL, NULL)) == -1) {
        perror("recvfrom failed");
        exit(EXIT_FAILURE);
    }

    printf("Response from server: %s\n", buffer);

    close(client_fd);

    return 0;
}
```