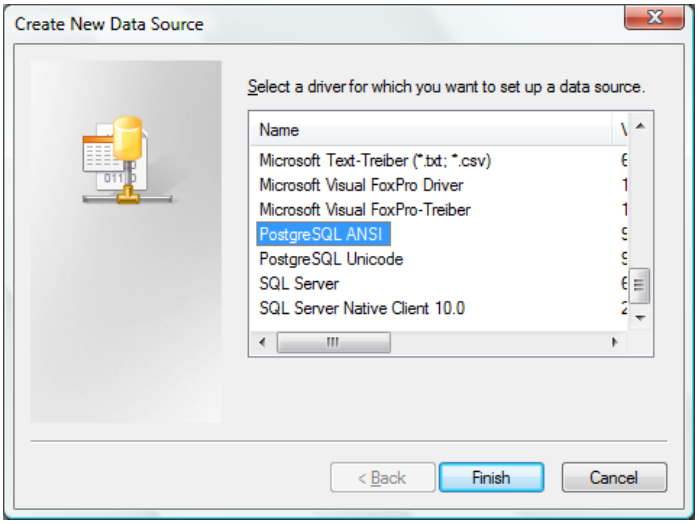
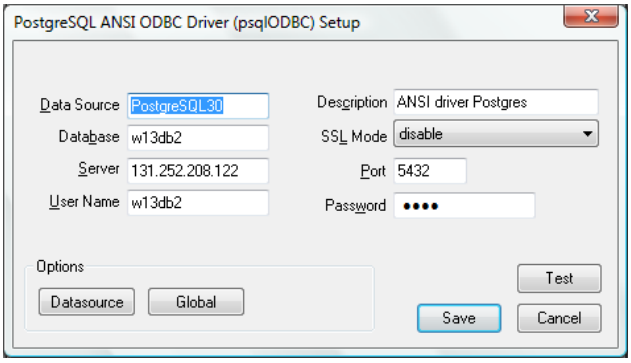
Steps for creating and executing embedded SQL in C:

1. Connect to VPN - It is needed because our database server does not accept any connections from outside the network
   http://cat.pdx.edu/windows/connecting-to-the-vpn-on-windows.html
   http://cat.pdx.edu/linux/connecting-to-the-vpn-on-linux-2.html
   http://cat.pdx.edu/mac/connecting-to-the-vpn-from-mac-2.html

2. Connect to PostgreSQL via ODBC:
   a. Download the PostgreSQL ODBC driver and install.
      http://www.postgresql.org/ftp/odbc/versions/
   b. Set up ODBC data source via the Windows data sources control panel. (this method might be different on different platforms.
      i. Go to "C:\Windows\SysWOW64\odbcad32.exe" (for 64 bit Windows system)
      ii. Click 'Add' button to add new data source. (A new window will display with all of the available ODBC drivers that are installed on your machine)
      iii. Select the appropriate option, click Finish.



      iv. Enter server, username, password etc and test the connection.



Reference links for ODBC data setup:
http://www.razorsql.com/docs/odbc_setup.html
http://www.easysoft.com/developer/interfaces/odbc/linux.html

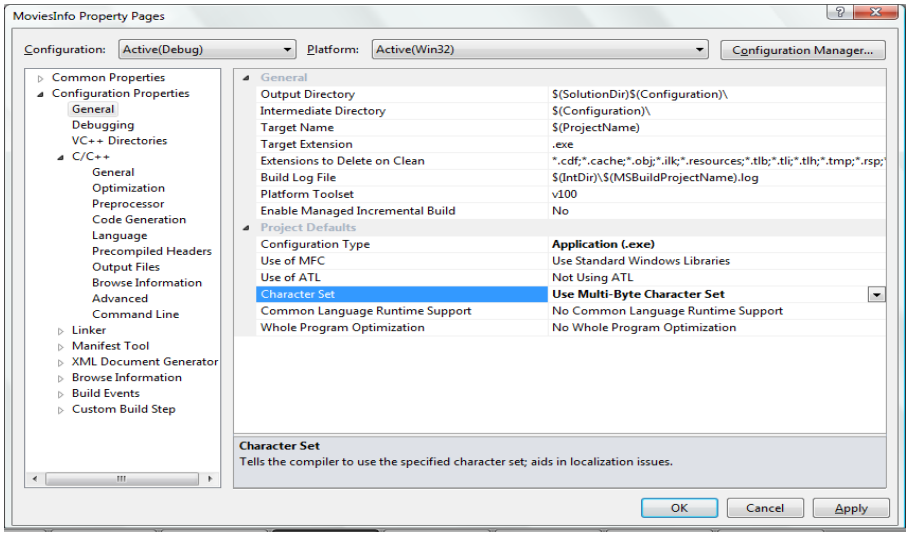
3. Now, write C program.

```
#include <stdio.h>
#include <windows.h>
#include <sql.h>
#include <sqlext.h>

void main() {
    SQLHENV env;
    SQLHDBC dbc;
    SQLHSTMT stmt;
    SQLRETURN ret; /* ODBC API return status */

    /* Allocate an environment handle */
    ret = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &env);
    printf("Return code for environment handle: %d\n", ret);

    /* We want ODBC 3 support */
    ret=SQLSetEnvAttr(env, SQL_ATTR_ODBC_VERSION, (void *) SQL_OV_ODBC3,
0);
    printf("Return code for set env attr: %d\n", ret);

    /* Allocate a connection handle */
    ret=SQLAllocHandle(SQL_HANDLE_DBC, env, &dbc);
    printf("Return code for dbc: %d\n", ret);

    /* Connect to the DSN */
    ret=SQLDriverConnect(dbc, NULL, (SQLCHAR *)"DSN=PostgreSQL30;",
SQL_NTS, NULL, 0, NULL, SQL_DRIVER_COMPLETE);
    printf("Return code for driver connect: %d\n", ret);

    /* Allocate a statement handle */
    ret=SQLAllocHandle(SQL_HANDLE_STMT, dbc, &stmt);
    printf("Return code for handle stmt: %d\n", ret);

    ret=SQLExecDirect(stmt, (SQLCHAR *)"INSERT INTO movies(title,year)
VALUES ('Hugo', 2012)", SQL_NTS);
    printf("Return code for execdirect: %d\n", ret);

    SQLSMALLINT year = 2012;
    SQLCHAR title[50] = "Life of Pie";
    SQLINTEGER lenyear=0, lentitle = SQL_NTS;

    /* Prepare */
    ret = SQLPrepare(stmt,(SQLCHAR *)"INSERT INTO movies(title,year)
VALUES (?, ?)", SQL_NTS);
    printf("ret prepare: %d\n", ret);

    /* Bind Parameters */
    ret = SQLBindParameter(stmt, 1, SQL_PARAM_INPUT, SQL_C_CHAR, SQL_CHAR,
50, 0, title, 0, NULL);
```

```
    printf("ret bind1: %d\n", ret);
    ret = SQLBindParameter(stmt, 2, SQL_PARAM_INPUT, SQL_C_SSHORT,
SQL_INTEGER, 0, 0, &year, 0, &lenyear);
    printf("ret bind2: %d\n", ret);

    /* Execute Query */
    ret = SQLExecute(stmt);
    printf("ret execute: %d\n", ret);

    /* disconnect from driver */
    SQLDisconnect(dbc);

    /* Free up allocated handles */
    SQLFreeHandle(SQL_HANDLE_STMT, stmt);
    SQLFreeHandle(SQL_HANDLE_DBC, dbc);
    SQLFreeHandle(SQL_HANDLE_ENV, env);

    printf("Done!\n");
    getchar();
}
```

NOTE:

1. In class, I had a problem with SQLDriverConnect. In Microsoft Visual Studio, I had to change the Character Set to 'Use Multi-Byte Character Set' to solve the problem.



2. PostgreSQL also provides a library for C - libpq. It has functions - PQconnectdb, PQexec, PQfinish etc. Instead of ODBC, you can also connect to the database using this library. Refer to - http://www.postgresql.org/docs/7.4/static/libpq.html