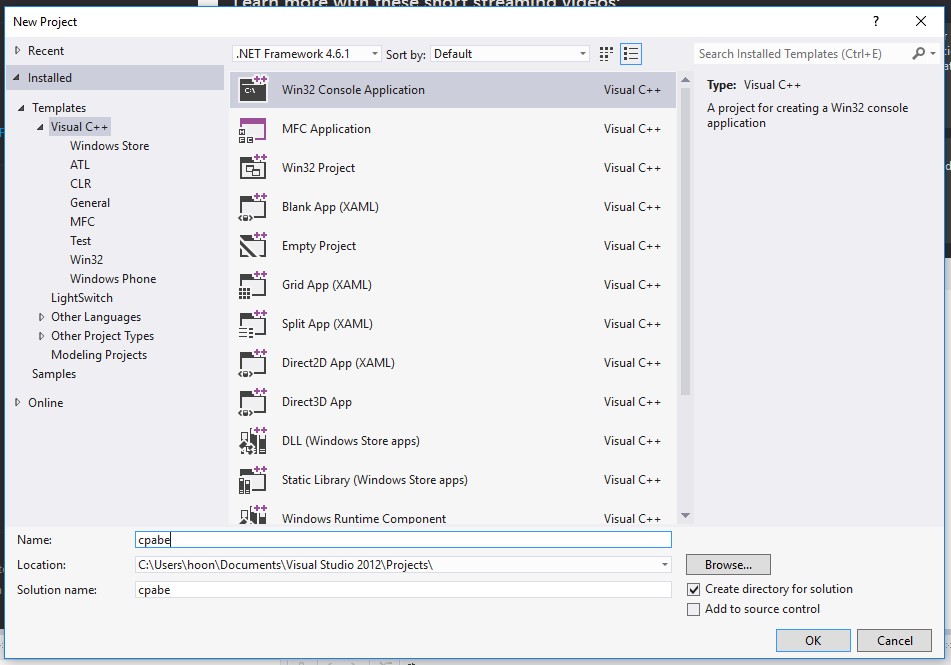
**Report for compiling the CPABE program:**

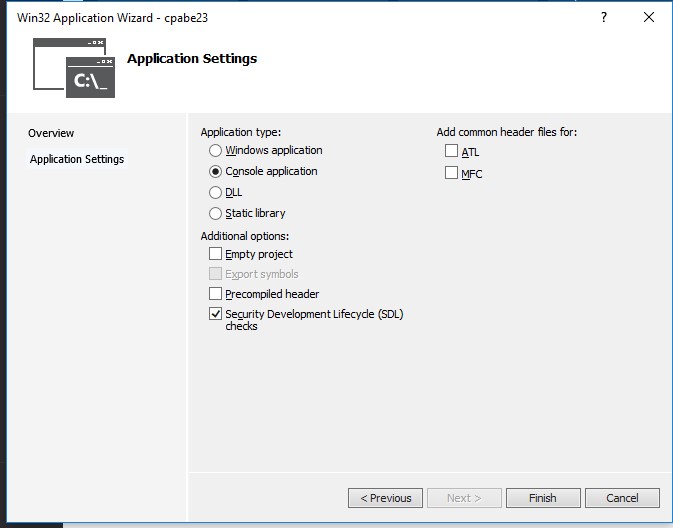
**Note: I have tried to answer all your comments.**

The program was compiled in Visual Studio 2012 and the operating system was Windows 10. In this report we will see how to compile the miracl cpabe program.

* Start with creating a new win 32 Console Application. Write project name “cpabe” and solution name “cpabe” and click ok. 

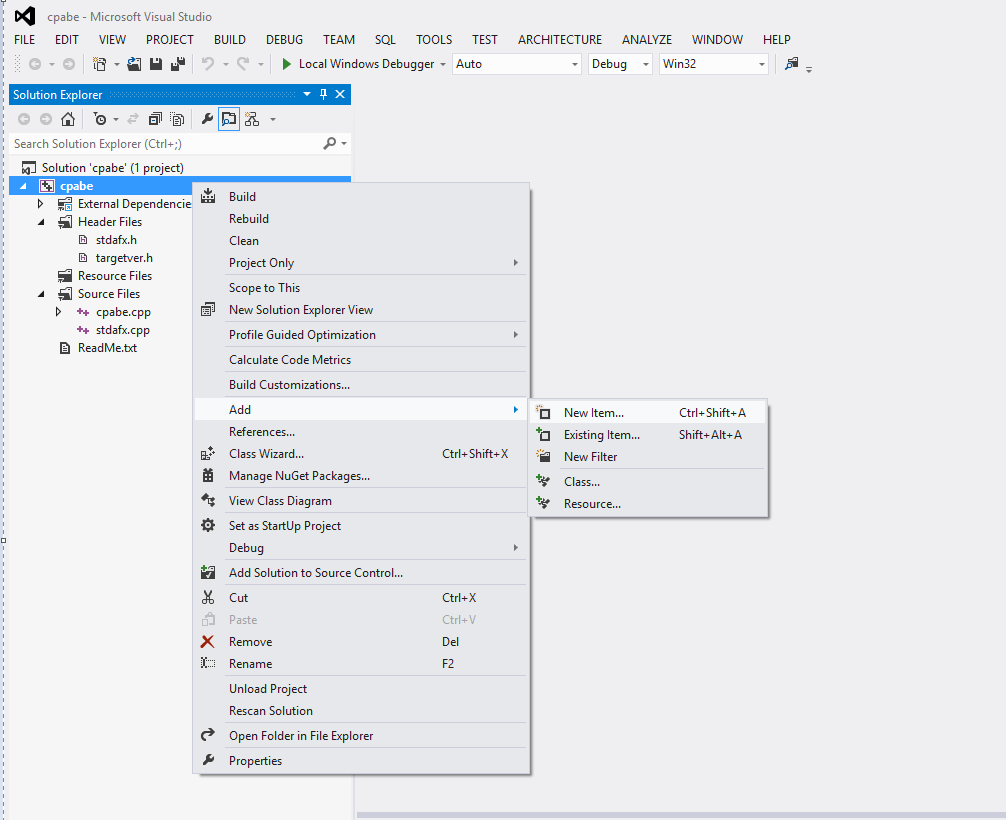
­­

* Now check Console application and Security Development Lifecycle, uncheck the “Precompiler header” in program settings and click finish. You can go to program setting from the left panel as show in the figure below.



* Right Click the project “cpabe” in the left panel and got to: Add🡪Existing Item.

Click “Existing Item”.



* Now add the files mentioned below against each type of pairing.
  + For MR\_PAIRING\_CP curve

cl /O2 /GX cpabe.cpp cp\_pair.cpp zzn2.cpp big.cpp zzn.cpp ecn.cpp.

# For MR\_PAIRING\_MNT curve

cl /O2 /GX cpabe.cpp mnt\_pair.cpp zzn6a.cpp ecn3.cpp zzn3.cpp zzn2.cpp big.cpp zzn.cpp ecn.cpp.

# For MR\_PAIRING\_BN curve

cl /O2 /GX cpabe.cpp bn\_pair.cpp zzn12a.cpp ecn2.cpp zzn4.cpp zzn2.cpp big.cpp zzn.cpp ecn.cpp.

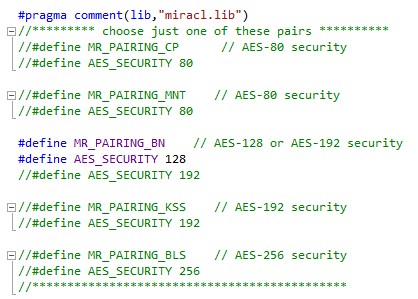
# For MR\_PAIRING\_BLS curve

cl /O2 /GX cpabe.cpp bls\_pair.cpp zzn24.cpp zzn8.cpp zzn4.cpp zzn2.cpp ecn4.cpp big.cpp zzn.cpp ecn.cpp.

# For MR\_PAIRING\_KSS curve

cl /O2 /GX cpabe.cpp kss\_pair.cpp zzn18.cpp zzn6.cpp ecn3.cpp zzn3.cpp big.cpp zzn.cpp ecn.cpp.

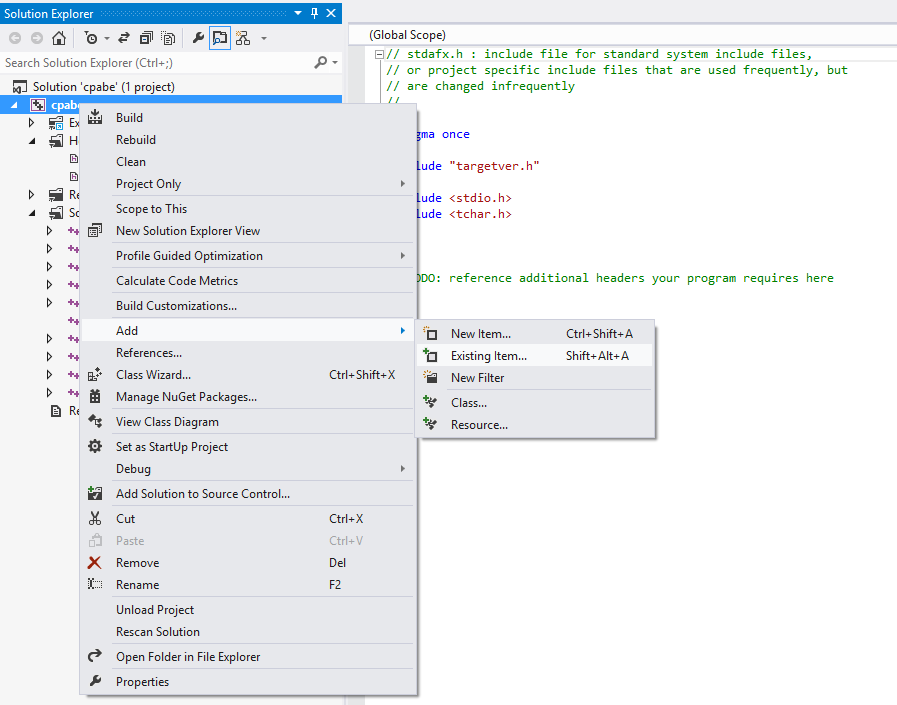
* To choose a pairing do the following:
  + Open the cpabe.cpp file. Go to the code section as shown in figure below.
  + Just uncomment the type of pairing and the security you want to select.
  + Like in this case the pairing that is chosen is “Pairing BN curve” and the security chosen is “AES Security 128”. As shown in the figure.
  + Note that “//” is used to comment in C++, to uncomment just remove the “//” from in front of the line you want to uncomment. Like in this case we have removed “//” at the starting of “#Define MR\_PAIRING\_BN” and “#Define AES\_SECURITY 128”



Note: Here you will notice that one “cpabe.cpp” file already exist before including the above mentioned files. So open the “cpabe.cpp” file you just included from Miracl distribution and copy all the code of this file into the already existing “cpabe.cpp” file and delete the newly added “cpabe.cpp” file.

* Right Click the project “cpabe” in the left panel and got to: Add🡪Existing Item. Click

“Existing Item”.



* Include the \*.h file for each type of pairing mentioned below.
* For MR\_PAIRING\_CP curve

cl /O2 /GX cpabe.h cp\_pair.h zzn2.h big.h zzn.h ecn.h miracl.lib

# For MR\_PAIRING\_MNT curve

cl /O2 /GX cpabe.h mnt\_pair.h zzn6a.h ecn3.h zzn3.h zzn2.h big.h zzn.h ecn.h.

# For MR\_PAIRING\_BN curve

cl /O2 /GX cpabe.h bn\_pair.h zzn12a.h ecn2.h zzn4.h zzn2.h big.h zzn.h ecn.h.

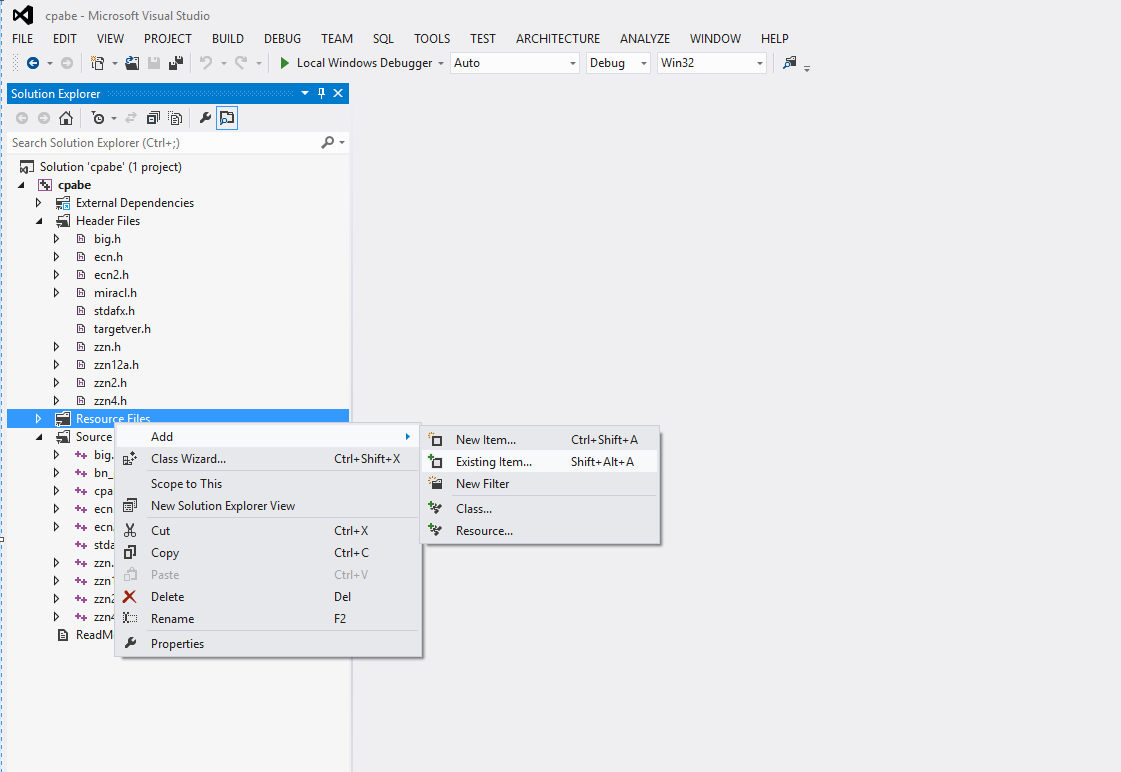
# For MR\_PAIRING\_KSS curve

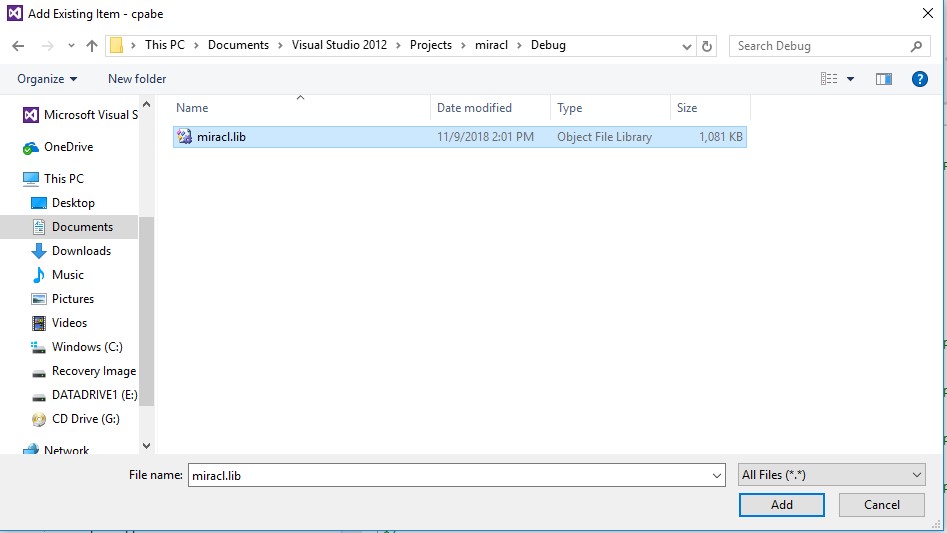
cl /O2 /GX cpabe.h kss\_pair.h zzn18.h zzn6.h ecn3.h zzn3.h big.h zzn.h ecn.h.

# For MR\_PAIRING\_BLS curve

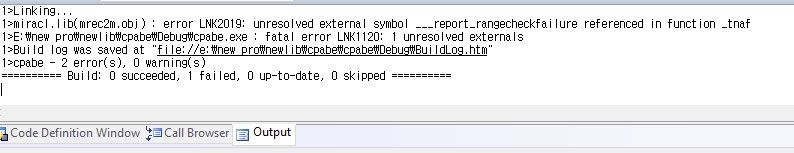
cl /O2 /GX cpabe.h bls\_pair.h zzn24.h zzn8.h zzn4.h zzn2.h ecn4.h big.h zzn.h ecn.h.

* Add the library file "miracl.lib” in resource folder. Right Click on resource folder and go to ADD🡪Existing Item.

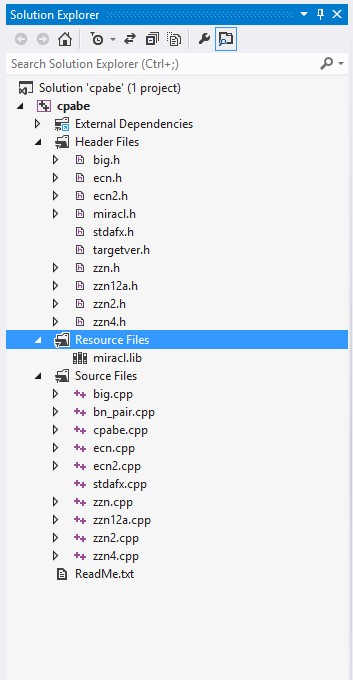




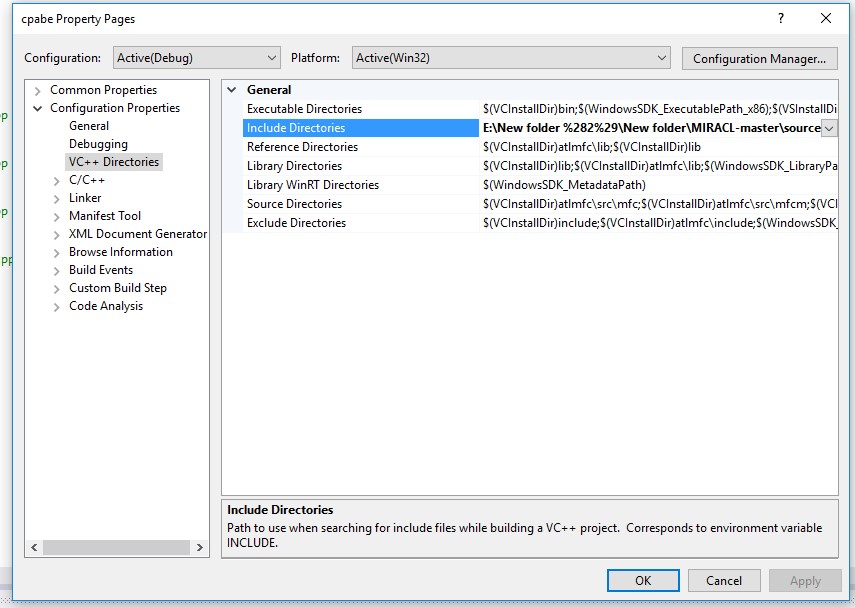
* We added “miracle.lib” to resource files to avoid the errors like error LNK2019, error LNK2001 as shown in figure below.



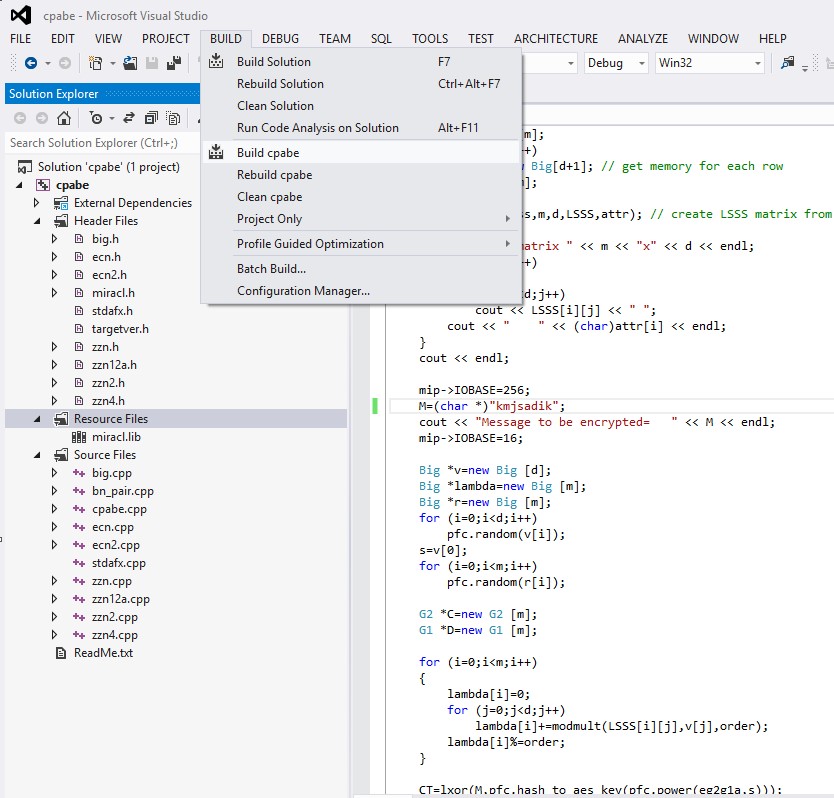
* After adding all the files as mentioned above the solution explorer should look as follow. Note that the following is for image for BN-Pairing.



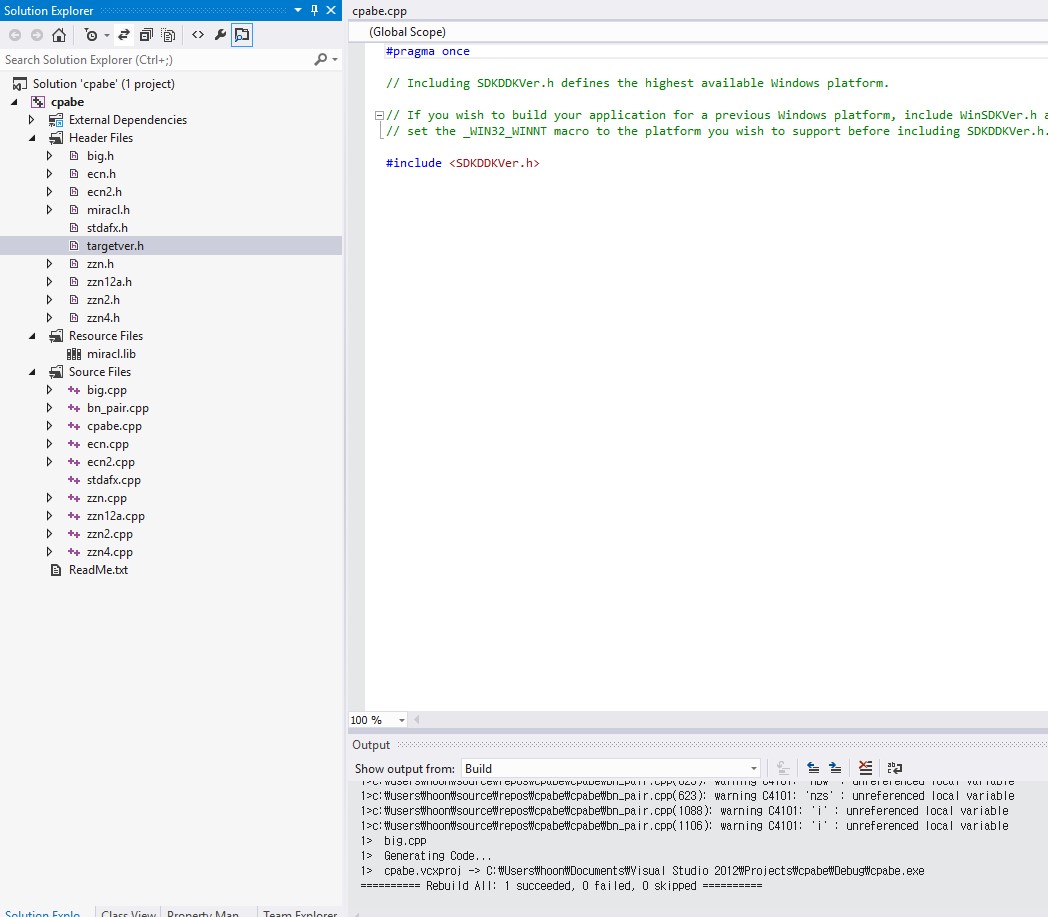
* Go to the properties🡪Configuration settings🡪VC++ directories🡪Include directories. Click on the drop down menu and select the “pairing” folder from miracl distribution.
* Go to the properties🡪Configuration settings🡪VC++ directories🡪Include directories. Click on the drop down menu and select the “include” folder from miracl distribution.
* Go to the properties🡪Configuration settings🡪VC++ directories🡪Include directories. Click on the drop down menu and select the “source” folder from miracl distribution. Then click on APPLY and OK.



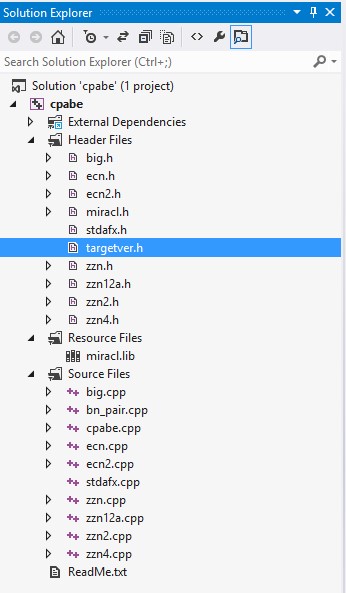
* Build the program from the Build Tab. Click Build cpabe.



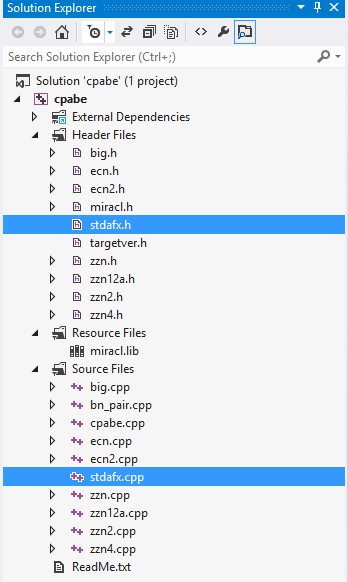
* The program should successfully build like below.



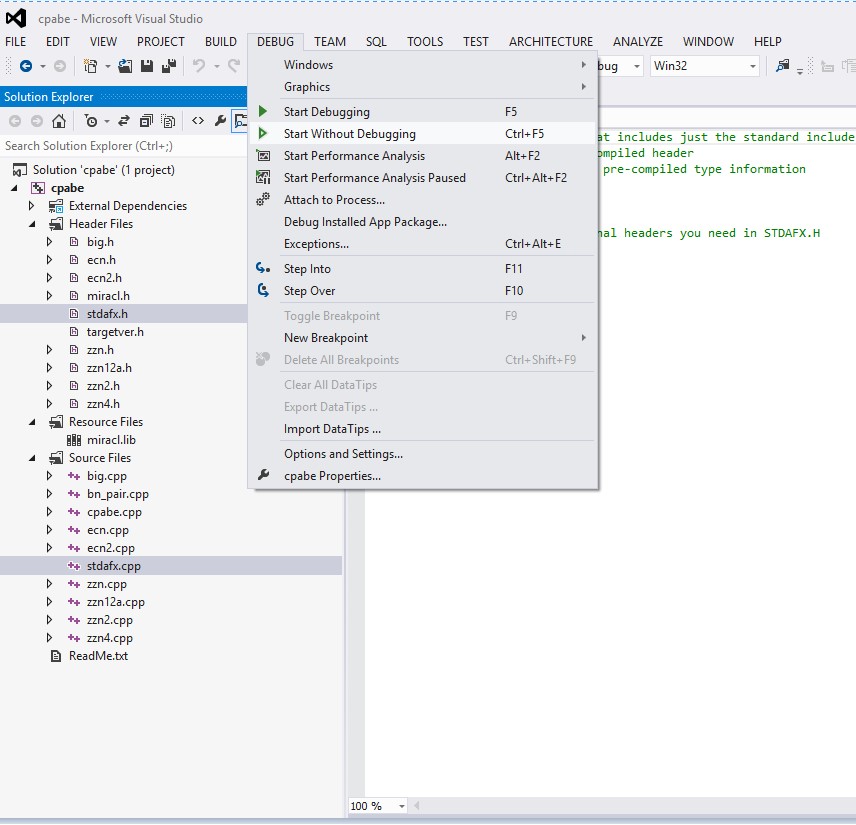
**Note**: From the solution explorer make sure that program has "targetver.h" in header files as show in below figure. As this file Include SDKDDKVer.h defines the highest available Windows platform. If you wish to build your application for a previous Windows platform, include WinSDKVer.h and set the \_WIN32\_WINNT macro to the platform you wish to support before including SDKDDKVer.h. “targetver.h” and “SDKDDKVer.h” are used to control what functions and constants are included into your code from the Windows headers, based on the OS. The “targetver.h” sets defaults to using the latest version of Windows unless thedefines are specified elsewhere. These two files (targetver.h and SDKDDKVer.h ) are auto generated when you create the project, you do not need to manually add them to the project.



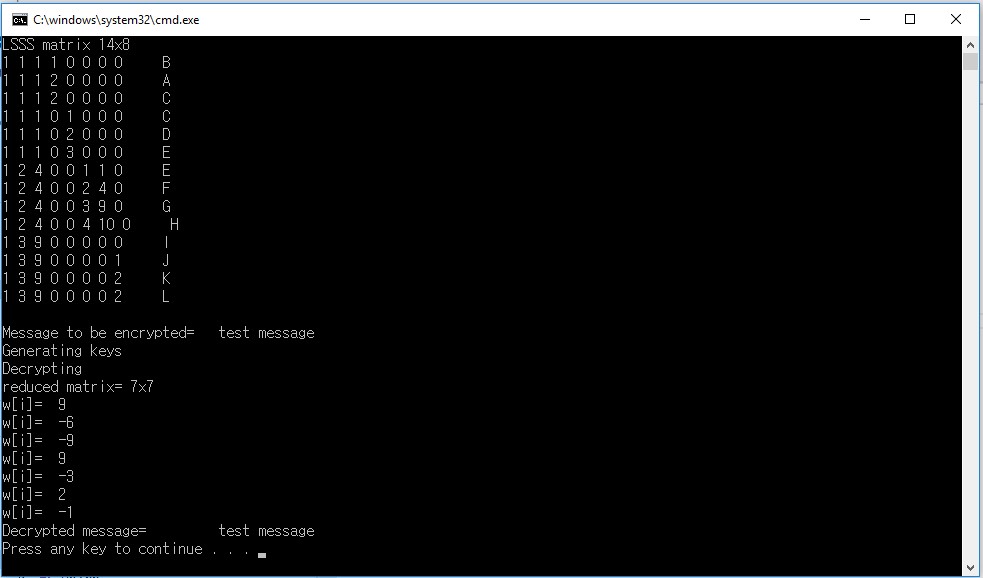
* Also make sure from the solution explorer that program has "stdafx.h" and "stdafx.cpp" as "stdafx.h" as show in figure below. It's a "precompiled header file" any headers you include in stdafx.h are pre-processed to save time during subsequent compilations. These two files (stdafx.cpp and stdafx.h) are auto generated when you create the project, you do not need to manually add them to the project or to create them.



* Now click on Debug and click on Start Without debugging. If you will click on start debugging the program will close soon after debugging, so you cannot see the program results.

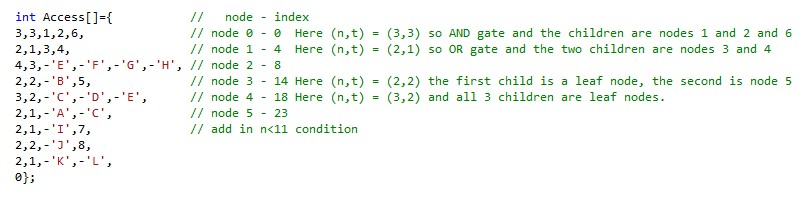


* When you will click on start without debugging after successfully building the program. The program will output the following, as shown in figure below.

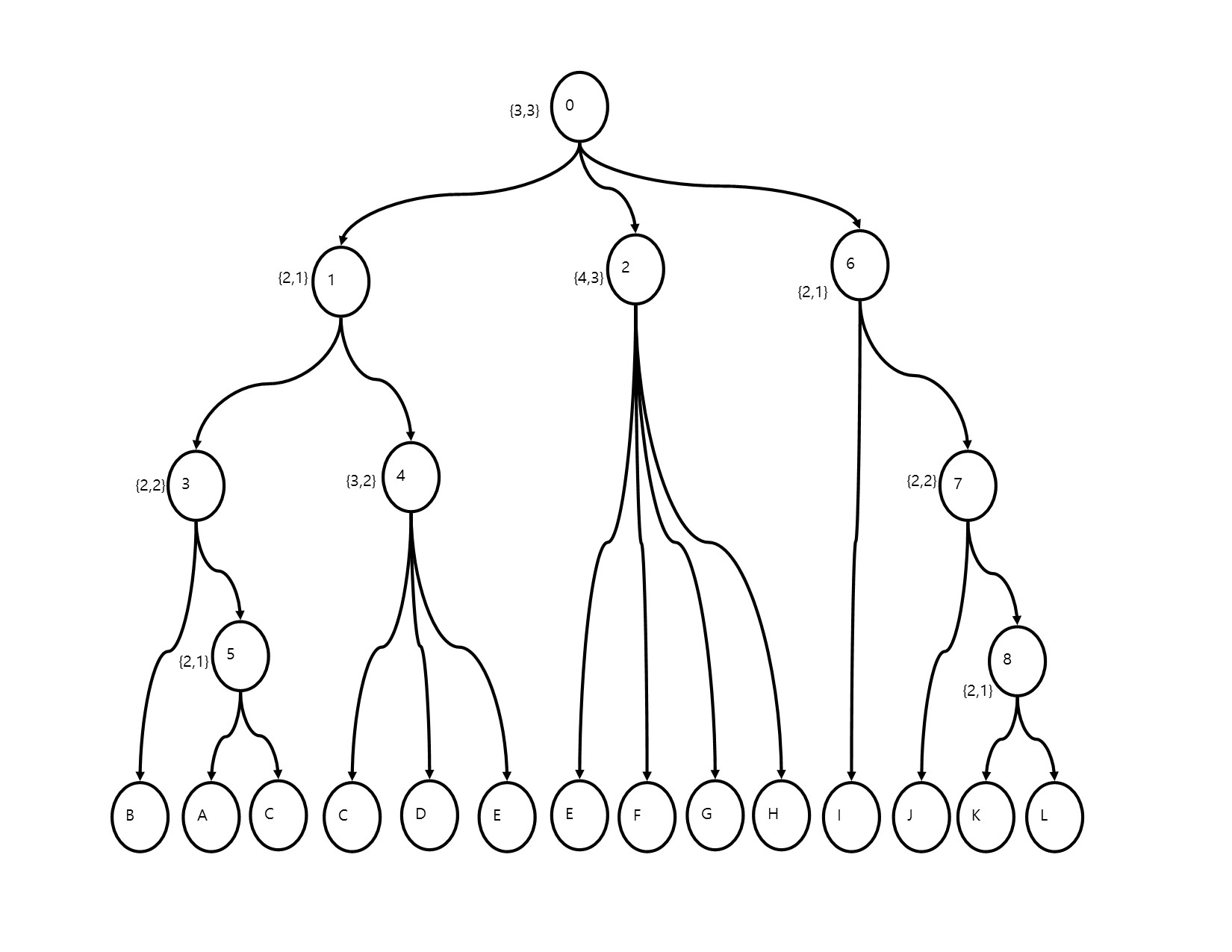


**Program Code**

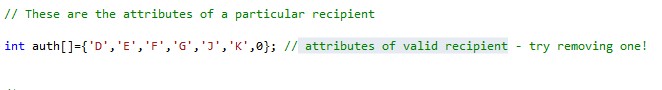
* The figure below shows the access structure of the scheme. The access structure is defined by an integer array “Access[]”. There are twelve attributes from A-L. The first two columns in a row shows the node number and the index (threshold)of the access structure.



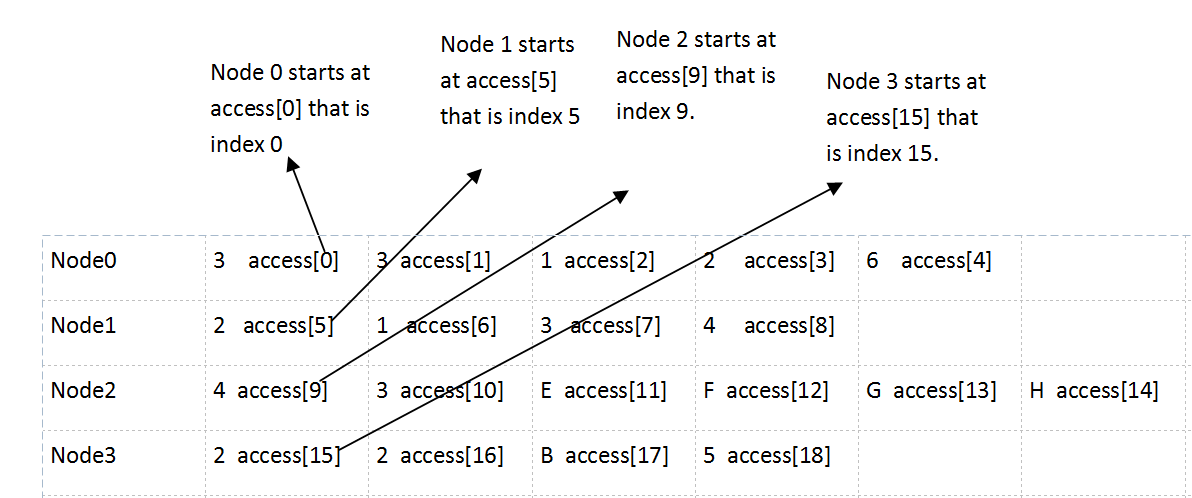
* The diagrammatic representation of the above access structure in the form of a tree is given below.

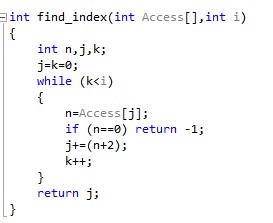


* The “auth[]” array in the figure below shows the attributes of a valid recipient.

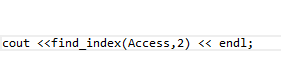


* **find\_index**: It is a function that takes the access structure and the node. It returns the index of required attribute. It checks the access structure nodes one by one, if we want to check the index of node “2” then it will give 9 as the index of node “2” as in the access structure array the index of node “2” is 9. Likewise the index of node 0 is 0. Index of node 1 is 5 and index of node 3 is 15 As shown in the figures below.





Calling of function “find\_index” by passing parameters access structure= “Access” and node= “2”:



Output of the function:



Calling of function “find\_index” by passing parameters access structure= “Access” and node= “3”:



Output of the function:



Calling of function “find\_index” by passing parameters access structure= “Access” and node= “0”:



Output of the function:



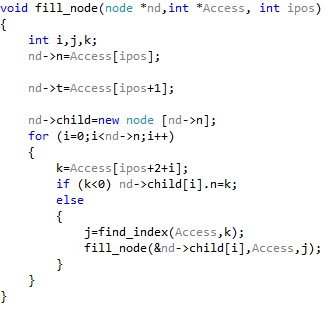
Calling of function “find\_index” by passing parameters access structure= “Access” and node= “1”:



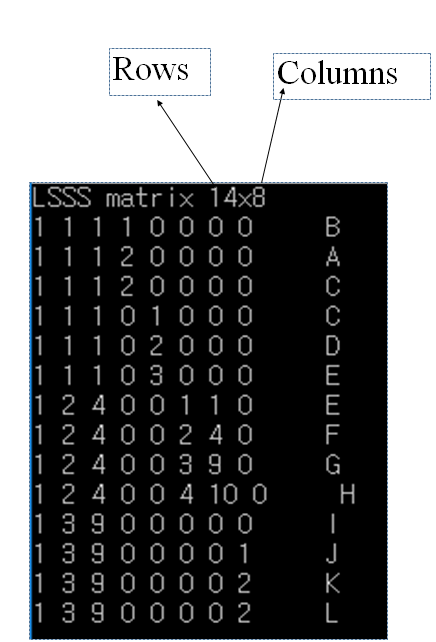
Output of the function:

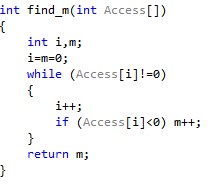


* **fill\_node** : It is a function to fill the Access Structure form the data. It takes the node pointer, the Access Structure to be filled and the data to be filled. Then it finds the given node in the access structure and fills it with the given data.



* **find\_m:** It is a function is used to find total number of leaf attributes. The number of rows in Access matrix. It simply checks the value of each node if the value of the node is less than zero then it is a leaf node otherwise not a leaf node. (We use matrix like this because in programming there are no trees and matrices. To implement trees and matrices, we use two dimensional arrays. LSSS matrices are easiest way to implement a tree)





Calling the function “find\_m” by passing parameter access structure= “Access”.

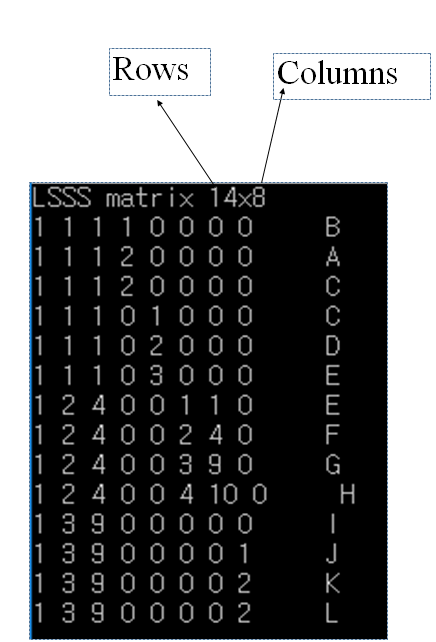


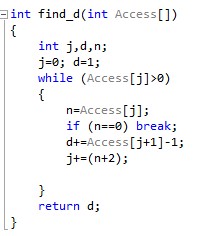
Output:



The output is 14, because there are total 14 attributes in the access structure “Access” that is A,B,C,D,E,F,G,H,I,J,K,L. As C and E are repeated in leaf nodes twice, so the total number of leaf nodes becomes 14.

* **find\_d:** It is a function to find number of columns in Access Structure.





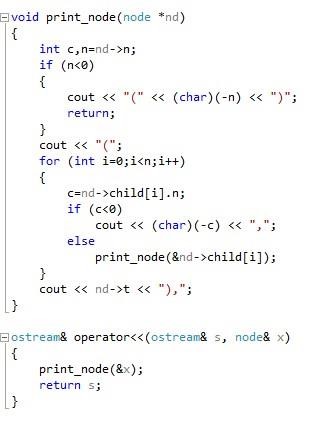
As show in the figure below:



Output:



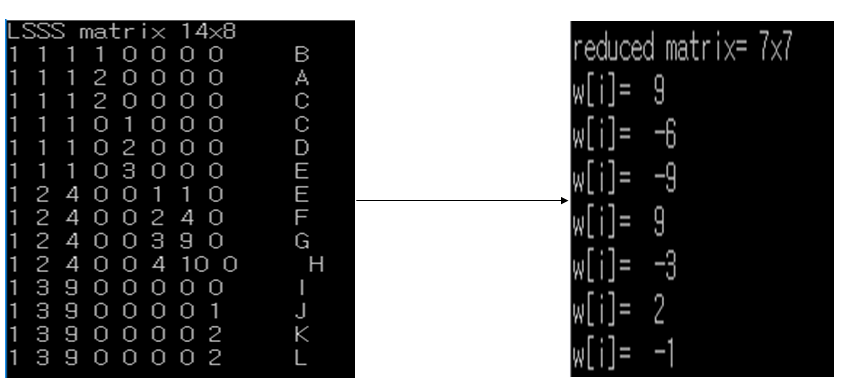
* **print\_node:** It is a function to traverse Access tree and print it to the console. It takes a node as input and gives the value of node as output. (This module cannot be run separately because it involves many classes and method to run properly. In order to run it we need to run the whole program otherwise it will prompt runtime errors)

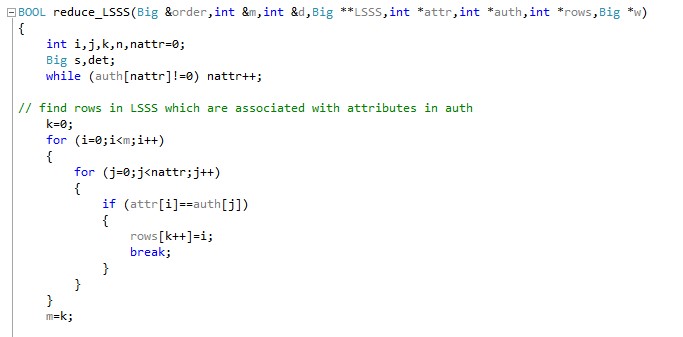


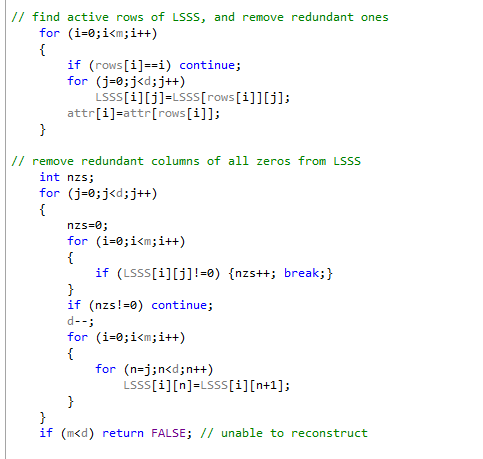
* **reduce-LSSS:**

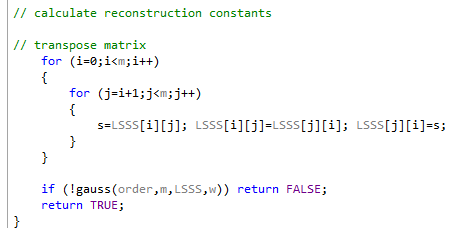
**(**The explanation of this function is in abstract level. So I cannot make it simpler, because to make to more simple we need to explain the detail of reduce echelon from and properties of matrices, which can be done by a mathematician, moreover I cannot write more simple English then this**).**

It is a function that is given set of attributes and LSSS (Linear Secret Sharing Scheme) matrix. The “reduce\_LSSS” function returns the rows and reconstruction constant numerators of “w”. Where “w” is an “n” element vector. Linear Secret Sharing Scheme (LSSS) matrices are commonly used for implementing monotone access structures in highly expressive Ciphertext-Policy Attribute-Based Encryption (CP-ABE) schemes. This LSSS matrix is used for specifying and enforcing the access policy of the ciphertext. The input parameter “order” is the pairing –friendly group order, “m” and “d” are public parameters defined in main function of the program. “m” contains value of “find\_m(Access)” and “d” contains value of “find\_d(Access)”. “\*\*LSSS” is pointer to LSSS matrix, “\*attr” is pointer to “attr[]” array, “\*auth” is pointer to “auth” array, “\*row” is pointer for number of leaf nodes. “\*w” is also pointer for number of leaf nodes but of type Big.





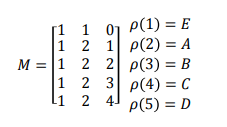




* **make\_LSSS:**

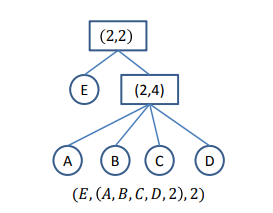
**(This is not mapping of above access structure. It is just another instance to make the mapping procedure understand. To understand the complete mapping one should read the whole paper provided in link below, because without reading paper this cannot be understand).**

It is a function makes LSSS(Linear Secret Sharing Scheme) matrix of size rows\*columns from Access description. “attr[i]” is an array, which contains attribute of each row (rows in the access structure “access”). You can find the complete Algorithm [here](http://eprint.iacr.org/2010/374). A Threshold-gate access tree and an equivalent LSSS matrix is show in figure below.

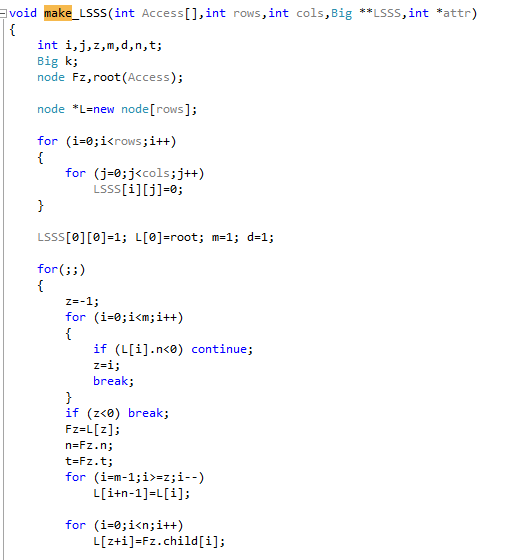


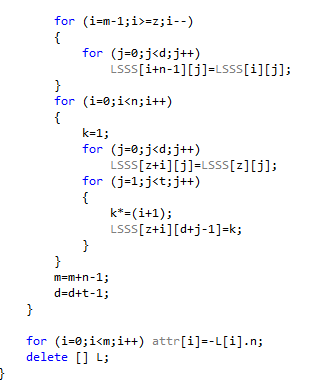
In the above figure an LSSS example (M, ρ) is given. M is a matrix and ρ is a functions, which maps the 1st, 2nd, 3rd, 4th and 5th row of M to attributes E, A, B, C and D respectively. Given an attribute set S, if and only if the rows of M labelled by the attributes in S include the vector (1, 0, …. , 0) in their span, the LSSS is said to be satisfied by S.

Threshold\_gate access tree, as shown in the figure below. Each non-leaf node is a threshold gate and each leaf is an attribute. Given an attribute set S, for a leaf node, if the associated attribute is in S, the leaf node is said to be satisfied by S. For a (t, n)-threshold node where n is the number of children and 1 ≤ t ≤ n a threshold value, if and only if at least “t” child nodes are satisfied, the threshold node is said to be satisfied, If and only if the root node of the access tree is satisfied. The access tree is said to be satisfied by S. Also note that when t = 1, a (t, n)-threshold gate is an OR gate and when t = n, it is an AND gate. The LSSS matrix represents an access policy that “an eligible user should have attribute E and in addition, at least two more attributes in {A, B, C, D}”. It can be expressed in a more comprehensible way, for example, E ∧((A∧B)∨(A∧C)∨(A∧D)∨(B ∧C)∨(B ∧D)∨(C ∧D)), which is a Boolean formula, or an access tree, namely (E,(A, B, C, D, 2), 2), as shown in the figure below, where each interior (i.e. non-leaf) node of the tree is a threshold gate and the leaf nodes are attributes



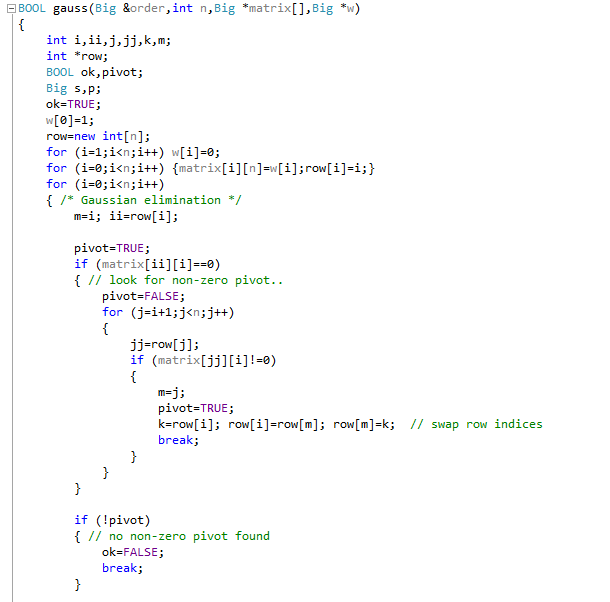
Program code for “make-LSSS” function.





* **gauss:** It is a function. Given Aw=I, where I = (1, 0, 0..., 0), the “gauss” function returns “w”. Where “A” is “nxn” matrix and “w” is an “n” element vector. We have used Gauss-Jordan elimination, because by using Gauss-Jordan elimination functions can be converted (converted means to change or to transform in to another form) into reduced echelon form (In [linear algebra](https://en.wikipedia.org/wiki/Linear_algebra), a [matrix](https://en.wikipedia.org/wiki/Matrix_(mathematics)) is in echelon form if it has the shape resulting from a [Gaussian elimination](https://en.wikipedia.org/wiki/Gaussian_elimination). Row echelon form means that Gaussian elimination has operated on the rows and column echelon form means that Gaussian elimination has operated on the columns. In other words, a matrix is in column echelon form if its [transpose](https://en.wikipedia.org/wiki/Transpose) is in row echelon form) without changing the access structure.

Program Code for “gauss” function:





(Yes, I remove that parts because they are part of reduce\_LSSS . They have been already mentioned in the section of reduce\_LSSS function. There is no need to mention it here again.)

* Below figure shows the main function of the “cpabe” program, where it initializes parameters for the construction of the CPABE scheme. The purpose for each parameter (variable) declaration is mentioned against each parameter (variable) in the figure.

