**Practical 1**

|  |  |  |
| --- | --- | --- |
| **Date:15/12/2022** | | |
| **Aim:** Apply attacks for cryptanalysis to decrypt the original message from a given cipher text using Caesar-Cipher. Soldier from field wants to send message to base. Implement the cipher to decrypt message. Decrypt message: KLURVKLPD for caesar cipher. | | |
| **Solution:**  #include <iostream>  #include <stdio.h>  using namespace std;  int main()  {      string str = "KLURVKLPD";      cout << "key :  Plaintext" << endl;      for (int i = 0; i <= 25; i++)      {          string res;          for (auto ch : str)          {              res += (ch - char(i) % char(26));          }          cout << i << "   :  " << res << endl;      }      return 0;  }  **Output:** | | |
| **Conclusion/Summary:** | | |
| **Student Signature & Date** | **Marks:** | **Evaluator Signature & Date** |

**Practical 2**

|  |  |  |
| --- | --- | --- |
| **Date:22/12/2022** | | |
| **Aim:** Soldier from field wants to send message to base. Implement the cipher to decrypt message. Using Playfair, Decrypt the message: BWPNRSMUALAW, Use key: pearlharbour | | |
| **Solution:**  #include <stdio.h>  #include <iostream>  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #define SIZE 30  void toLowerCase(char plain[], int ps)  {      for (int i = 0; i < ps; i++)      {          if (plain[i] > 64 && plain[i] < 91)              plain[i] += 32;      }  }  int removeSpaces(char \*plain, int ps)  {      int count = 0;      for (int i = 0; i < ps; i++)          if (plain[i] != ' ')              plain[count++] = plain[i];      plain[count] = '\0';      return count;  }  void generateKeyTable(char key[], int ks, char keyT[5][5])  {      int i, j, k, flag = 0, \*dicty;      dicty = (int \*)calloc(26, sizeof(int));      for (i = 0; i < ks; i++)      {          if (key[i] != 'j')              dicty[key[i] - 97] = 2;      }      dicty['j' - 97] = 1;      i = 0;      j = 0;      for (k = 0; k < ks; k++)      {          if (dicty[key[k] - 97] == 2)          {              dicty[key[k] - 97] -= 1;              keyT[i][j] = key[k];              j++;              if (j == 5)              {                  i++;                  j = 0;              }          }      }      for (k = 0; k < 26; k++)      {          if (dicty[k] == 0)          {              keyT[i][j] = (char)(k + 97);              j++;              if (j == 5)              {                  i++;                  j = 0;              }          }      }  }  void search(char keyT[5][5], char a, char b, int arr[])  {      int i, j;      if (a == 'j')          a = 'i';      else if (b == 'j')          b = 'i';      for (i = 0; i < 5; i++)      {          for (j = 0; j < 5; j++)          {              if (keyT[i][j] == a)              {                  arr[0] = i;                  arr[1] = j;              }              else if (keyT[i][j] == b)              {                  arr[2] = i;                  arr[3] = j;              }          }      }  }  int mod5(int a)  {      if (a < 0)          a += 5;      return (a % 5);  }  void decrypt(char str[], char keyT[5][5], int ps)  {      int i, a[4];      for (i = 0; i < ps; i += 2)      {          search(keyT, str[i], str[i + 1], a);          if (a[0] == a[2])          {              str[i] = keyT[a[0]][mod5(a[1] - 1)];              str[i + 1] = keyT[a[0]][mod5(a[3] - 1)];          }          else if (a[1] == a[3])          {              str[i] = keyT[mod5(a[0] - 1)][a[1]];              str[i + 1] = keyT[mod5(a[2] - 1)][a[1]];          }          else          {              str[i] = keyT[a[0]][a[3]];              str[i + 1] = keyT[a[2]][a[1]];          }      }  }  void decryptByPlayfairCipher(char str[], char key[])  {      char ps, ks, keyT[5][5];      // Key      ks = strlen(key);      ks = removeSpaces(key, ks);      toLowerCase(key, ks);      // ciphertext      ps = strlen(str);      toLowerCase(str, ps);      ps = removeSpaces(str, ps);      generateKeyTable(key, ks, keyT);      decrypt(str, keyT, ps);  }  int main()  {      char str[SIZE], key[SIZE];      strcpy(key, "pearlharbour");      printf("Key text: %s\n", key);      strcpy(str, "BWPNRSMUALAW");      printf("Plain text: %s\n", str);      decryptByPlayfairCipher(str, key);      printf("Deciphered text: %s\n", str);      return 0;  }  **Output:** | | |
| **Conclusion/Summary:** | | |
| **Student Signature & Date** | **Marks:** | **Evaluator Signature & Date** |

**Practical 3**

|  |  |  |
| --- | --- | --- |
| **Date:29/12/2022** | | |
| **Aim:** The Rail Fence Cipher was invented in ancient times. It was used by the Greeks, who created a special tool, called scytale, to make message encryption and decryption easier. The letters are arranged in a way which is similar to the shape of the top edge of the rail fence. If king Leonidas want to send message to Sparta as “300 achieved glory at hot gate, unite for Greece” then what will be cipher text when it is encrypted using 3 rows. Also implement decryption of message. | | |
| **Solution:**  #include <stdio.h>  #include <iostream>  #include <string>  using namespace std;  string encryptRailFence(string text, int key)  {      char rail[key][(text.length())];      for (int i = 0; i < key; i++)          for (int j = 0; j < text.length(); j++)              rail[i][j] = '\n';      bool dir\_down = false;      int row = 0, col = 0;      for (int i = 0; i < text.length(); i++)      {          if (row == 0 || row == key - 1)              dir\_down = !dir\_down;          rail[row][col++] = text[i];          dir\_down ? row++ : row--;      }      string result;      for (int i = 0; i < key; i++)          for (int j = 0; j < text.length(); j++)              if (rail[i][j] != '\n')                  result.push\_back(rail[i][j]);      return result;  }  string decryptRailFence(string cipher, int key)  {      char rail[key][cipher.length()];      for (int i = 0; i < key; i++)          for (int j = 0; j < cipher.length(); j++)              rail[i][j] = '\n';      bool dir\_down;      int row = 0, col = 0;      for (int i = 0; i < cipher.length(); i++)      {          if (row == 0)              dir\_down = true;          if (row == key - 1)              dir\_down = false;          rail[row][col++] = '\*';          dir\_down ? row++ : row--;      }      int index = 0;      for (int i = 0; i < key; i++)      {          for (int j = 0; j < cipher.length(); j++)          {              if (rail[i][j] == '\*' && index < cipher.length())                  rail[i][j] = cipher[index++];          }      }      string result;      row = 0, col = 0;      for (int i = 0; i < cipher.length(); i++)      {          if (row == 0)              dir\_down = true;          if (row == key - 1)              dir\_down = false;          if (rail[row][col] != '\*')              result.push\_back(rail[row][col++]);          dir\_down ? row++ : row--;      }      return result;  }  int main()  {      string simpleText = "300 achieved glory at hot gate, unite for Greece";      string cipherText = encryptRailFence(simpleText, 3);      cout << "Cipher Text: " << cipherText << endl;      cout << "Simple Text: " << decryptRailFence(cipherText, 3) << endl;      cout << "Made by: 20DCS103 - Rushik Rathod";      return 0;  }  **Output:** | | |
| **Conclusion/Summary:** | | |
| **Student Signature & Date** | **Marks:** | **Evaluator Signature & Date** |

**Practical 4**

|  |  |  |
| --- | --- | --- |
| **Date:05/01/2023** | | |
| **Aim:** The transmission of information needs to be secure over the communication channel and the data has to be confidential. Study and implement the practical approach for  Steganography.   * Using OpenPuff Tool | | |
| **Steps to Hide:**   1. **After unzipping OpenPuff into a directory.**      1. **Click on OpenPuff.**      1. **Create hidden and original folders as shown below.**      1. **Create a new text file and add an image into folder named original.**      1. **Now, click on Hide in OpenPuff.**   **Enter password 🡪 Select the target textfile from the original folder 🡪 Select Carrier image from the original folded 🡪 Select the type of the image.**     1. **On clicking Hide Data 🡪 Select the output directory named hidden and after confirming, carrier is processed.**      1. **Viewing Report generated for Carrier.**      1. **Viewing Carrier Generated file in Directory.**     **Steps to Unhide:**   1. **Open OpenPuff 🡪 Select unhide.**      1. **Enter password 🡪 Select output directory hidden 🡪 Click OK**      1. **Unhide Report.**      1. **Checking output directory ‘hidden’ and a text file.** | | |
| **Conclusion/Summary:** | | |
| **Student Signature & Date** | **Marks:** | **Evaluator Signature & Date** |

**Practical 5**

|  |  |  |
| --- | --- | --- |
| **Date: / /2023** | | |
| **Aim:** Implement GPG for windows. | | |
| **Solution:**  **Steps:**   1. **Download GnuPG 3.0.2**      1. **Install GnuPG, open the software and click ‘Generate key now’.**      1. **Sender side.**   https://lh3.googleusercontent.com/QaZ2w4XBt0FxqrOXOIxrekEzfgJg2Zqdt7Hno0-XSiFcAazFDdrbQ2BFTihvKNBAXgkrWTAnuvO9Jf6x9Yi8uleZWqWvi43FUj4UBID9fXkutJvCq9OiuL4XPIEd4sVX5TfwVO24DKjsQ_s6DxXFtw0  https://lh3.googleusercontent.com/GksXiplbxmFHz0p-kVKGH1EGBzLoXdO_oBe1kRWkGGPUyTYt__rLrfjz_zpLKPJws3u3xZZUOnjPsao7AAZP377YEMoDtvZrzUG8o9Xz-cIswa7vP00yh1bwV4lm1ZgEusqPadCuG92oPjKAM23-yTY   1. **Receiver side.**   https://lh3.googleusercontent.com/O92K4rvbOrrgIp_Xqu_pr4RLihfjX34PZ7RhEuVHCfrtO-sVeEmZQ2CC9H8e2TVcEe1JhRv0yDFeF5289MEyRK4rYHpH3Y1npchY4FvOvZAhMfiFhKimg89P-Wt2vAbFC19fYOslHX2B5gSpCuohyQk  **https://lh5.googleusercontent.com/cmNUAvPY2Gw_pBEJVgikD_VrwxZfvmE16Z7FKYHw66fV2sKnc2zjlxqreKF6GhKP3n0i2FQZtKoUH4-i2Y5PwqWbPUgTf9406Zc2dBqbYBo47GhwXNUcHCphLcIfbfwhKw8_lDxULQg0mOVsF-4_jHo**   1. **Then create the receiver’s & sender’s public and private key respectively.**   **https://lh6.googleusercontent.com/dcqrIFj3h13hF_wop3fPpzvoh0Lfo3MvfMAbTVGVge9KsBHa8T9vSGpJauBmLoaGJPZVsDuTifs7cfymwRUgLepBsKMlXibvUNNNOA3fqpIRyHX9SaaCVfMhwlaSzaBNW1U2pqlcX-L5XssbYV31Gn0**   1. **Create the backup file for your key on the desktop.**   **F:\B.Tech. Sem 6\CRNS Practical\p5\3.jpg**  **Enter Passphrase and click OK.**  **https://lh4.googleusercontent.com/wTOwBwS7geVqEP8rg26eMrbB7GN8lCkan_bSHOLl2Rpj2NYaL8jvoBH9UnngkBpweuAv5vr1ck5GGBxydH3GyAZzoljhaJik4_SpeDmskycd15kANBR0jBPw8V6mji4n9VKOD8Ni8FuFmXhdJ8l2AV4**  **F:\B.Tech. Sem 6\CRNS Practical\p5\4.jpg**   1. **Then open the file where key is stored and edit out the private key of receiver and then send the remaining asc file (which contains public key of receiver) to the sender. Import the key of receiver on the sender’s side to encrypt the message.**   https://lh6.googleusercontent.com/msTwtNQUGw2yNNoArMZTgGGYWThXaxLwPmCfAQ6wOU0OYNgFyD1FQjZH1J8G0nARAQIAACUKRQaGnjXmE-ldXK8WWEyfnCKAGSL0mZXAN4MI8cXaJUJ7Pubc51RJ7dmjGanj1UkeHLL2F_3IsVOKAoU  https://lh4.googleusercontent.com/NIOryxgrxCAiIWwEOHmyZweSZCjyt0JGZkphGpDwCW-u3IBTxDvDALAScRUmiRyR7vktPPHe8ENx6zVejMpbg23Bs2Khgfq5W80VzHGwJiXbWErKymZLDyiW02zWNxMDLsvpUL7aemsswjpZs4e7EE8  https://lh4.googleusercontent.com/YaATph8aos2XgA2kMloZDHq1pkzlo6KUWgCoeY9vFMQcc7WNo6JkzHjwm09fkQMa-QotCEUPZs82J-6dAs1XmLDR00NutejuXtq0Ph8TDB2N8ajcvRNcYJpdGTPnlMZYhH5ZZ9WYgObBRwiVxmWJ8Yg   1. **Here the message is encrypted using the receiver’s public key.**   https://lh5.googleusercontent.com/SeMmdWj3wdeOJISzPqRvlx691nkCQ88GJCMnH6O1SwoinfhS4KvcWOWlWnCpeEVBHMVzwvdWr-6ANoX4pSmrmxfrZXFzI0HmIHh1jsqwYklWIpAm6Dz-OfsYefho3SQuRP7XTbJq5C6VRMy_kf-4PA0   1. **Decrypt the message using receiver’s private and public keys on receiver’s side.**   **https://lh3.googleusercontent.com/MfHEMP1p5QyTUUaKmBrpWVpP6-vaPXmH_zD7m99KPjz364f8DeWRWAvAFb5CZn177IXyGc1Y8-9c0-LXea1140HK6bYGYVlc_xxoRaCrKzjNTGfEWDuDBHLLnF6VjMnnNElO2bYSX1w342fg8_FqOnU**   1. **Message is decrypted and original message is shown in the figure below.**   **https://lh6.googleusercontent.com/RZ4jJuGUB2tmEim0QEPH-90fH26rZGLaYDnK1HEKvfNoIS2UahDTlinpN88qrRBsM7Uup-YMIWxjeguX9YjrbR1oNl4IveTp7BJ7CFD3gWrPXbPADw_APVJXNEeiSINnGfLsd6JznOx12qsWGm03Pcw** | | |
| **Conclusion/Summary:** | | |
| **Student Signature & Date** | **Marks:** | **Evaluator Signature & Date** |

**Practical 6**

|  |  |  |
| --- | --- | --- |
| **Date: / /2023** | | |
| **Aim:** Configure Nmap (Network mapping tool) on Linux/Windows. Explore the various command for scanning your host/ips, ports and various services running on port. Prepare the document of at least 25 Nmap commands. Use the Nmap script and launch the DoS attack by flooding the packages in regular interval. | | |
| **Solution:**  **Steps:**   1. **To scan a System with its IP address.**      1. **To scan using “-v” option.**      1. **To scan to detect firewall settings.**      1. **To scan multiple hosts**      1. **To identify hostnames**      1. **To scan from a file**      1. **To get some help.**      1. **Here -A indicates aggressive, it will give us extra information, like OS detection (-O), version detection, script scanning (-sC), and traceroute (–traceroute). It even provides a lot of valuable information about the host.**      1. **Using this command we can discover the target hosting service or identify additional targets according to our needs for quickly tracing the path.**      1. **Here it will display the operating system where the domain or ip address is running, but will not display the exact operating system available on the computer. It will display only the chance of operating system available in the computer. The command will just guess the running operating system (OS) on the host.** | | |
| **Conclusion/Summary:** | | |
| **Student Signature & Date** | **Marks:** | **Evaluator Signature & Date** |

**Practical 7**

|  |  |  |
| --- | --- | --- |
| **Date: / /2023** | | |
| **Aim:** Perform Port Scanning, File Transfer, Client-server chat and Basic Webserver implementation using netcat. Find the service running on the particular port using netcat. | | |
| **Solution:**  **Port Scanning:**   * This is useful to know which ports are open and running services on a target machine. * The -z flag can be used to tell nc to report open ports, rather than initiate a connection.     **File Transfer:**   * The nc ( netcat ) command can be used to transfer arbitrary data over the network. * It represents a quick way for Linux administrators to transfer data without the need for an additional data transfer services such as FTP, HTTP, SCP etc.       **Client-Server Chat:**   * To create a simple chat we need two instances of netcat, one to listen for incoming connections (the server) and another one to start the connection.       **Basic Webserver Implementation:**   * The netcat tool nc can operate as a TCP client. Because HTTP works over TCP, nc can be used as an HTTP server! * Because nc is a UNIX tool, we can use it to make custom web servers: servers which return any HTTP headers you want, servers which return the response very slowly, servers which return invalid HTTP, etc. | | |
| **Conclusion/Summary:** | | |
| **Student Signature & Date** | **Marks:** | **Evaluator Signature & Date** |

**Practical 8**

|  |  |  |
| --- | --- | --- |
| **Date: / /2023** | | |
| **Aim:** Bob is going to send his encrypted file using public key shared by Alice using Public-key infrastructure. Alice will decrypt the file by using her private key and ensure the confidentiality. Implement the given scenario using RSA algorithm. | | |
| **Solution:**  **Code:**  #include<iostream>  #include<math.h>  using namespace std;  int gcd(int a, int b) {  int t;  while(1) {  t= a%b;  if(t==0)  return b;  a = b;  b = t;  }  }  int main() {  double p = 13;  double q = 11;  double n=p\*q;  double track;  double phi= (p-1)\*(q-1);  double e=7;  while(e<phi) {  track = gcd(e,phi);  if(track==1)  break;  else  e++;  }  double d1=1/e;  double d=fmod(d1,phi);  double message = 9;  double c = pow(message,e);  double m = pow(c,d);  c=fmod(c,n);  m=fmod(m,n);  cout<<"Original Message = "<<message;  cout<<"\n"<<"p = "<<p;  cout<<"\n"<<"q = "<<q;  cout<<"\n"<<"n = pq = "<<n;  cout<<"\n"<<"phi = "<<phi;  cout<<"\n"<<"e = "<<e;  cout<<"\n"<<"d = "<<d;  cout<<"\n"<<"Encrypted message = "<<c;  cout<<"\n"<<"Decrypted message = "<<m;    return 0;  }  **Output:** | | |
| **Conclusion/Summary:** | | |
| **Student Signature & Date** | **Marks:** | **Evaluator Signature & Date** |

**Practical 9**

|  |  |  |
| --- | --- | --- |
| **Date: / /2023** | | |
| **Aim:** In computers, Foot printing is the process of accumulating data regarding a specific network environment, usually for the purpose of finding ways to intrude into the environment. Foot printing can reveal system vulnerabilities and improve the ease with which they can be exploited.  Use the given approach to implement Footprinting: Gathering Target Information making use of following tools:   * Dmitry – Deepmagic * Whatweb | | |
| **Solution:**  **Dmitry – Deepmagic:**  Dmitry is a free and open-source tool available on GitHub. The tool is used for information gathering. You can download the tool and install in your Kali Linux. Dmitry stands for DeepMagic Information Gathering Tool. It’s a command-line tool Using Dmitry tool You can collect information about the target, this information can be used for social engineering attacks. It can be used to gather a number of valuable pieces of information  Usages of Dmitry Tool :   * Dmitry Tool can be used to search subdomains of the target. * Dmitry Tool can be used to find open ports of the target system. * Dmitry Tool can be used to perform TCP scan. * Dmitry Tool can be used with netcraft service to get the target information such as operating system, web server details, web host details, hosting service details, etc. * Dmitry Tool can be used with whois service to get the target information such as registered domain, name, address, the contact information of the person who registered it. * Dmitry Tool can be used to get email addresses that are associated with the domain of the target.   **F:\B.Tech. Sem 6\CRNS Practical\p9\Capture1.JPG**  **F:\B.Tech. Sem 6\CRNS Practical\p9\Capture2.JPG**  **F:\B.Tech. Sem 6\CRNS Practical\p9\Capture3.JPG**  **F:\B.Tech. Sem 6\CRNS Practical\p9\Capture4.JPG**  **F:\B.Tech. Sem 6\CRNS Practical\p9\Capture5.JPG**  **F:\B.Tech. Sem 6\CRNS Practical\p9\Capture6.JPG**  **Whatweb:**  Whatweb is a free and open-source tool available on GitHub. Whatweb is a scanner written in the Ruby language. This tool can identify and recognize all the web technologies available on the target website. This tool can identify technologies used by websites such as blogging, content management system, all JavaScript libraries. Whatweb contains more than 180 modules. each module is responsible for grabbing particular information from the target website. Whatweb works as an information-gathering tool and can identify all the email addresses, SQL errors, technology used in the website.  **F:\B.Tech. Sem 6\CRNS Practical\p9\Capture0.JPG** | | |
| **Conclusion/Summary:** | | |
| **Student Signature & Date** | **Marks:** | **Evaluator Signature & Date** |

**Practical 10**

|  |  |  |
| --- | --- | --- |
| **Date: / /2023** | | |
| **Aim:** Find out Web Application Vulnerability using OWASP-ZAP tool. | | |
| **Solution:**  **What is ZAP?**  OWASP ZAP is an open-source web application security scanner. It is intended to be used by both those new to application security as well as professional penetration testers. It is one of the most active Open Web Application Security Project projects and has been given Flagship status.   * OWASP stands for “Open Web Application Security Project”. * It is an open, online community that creates methodologies, tools, technologies and guidance on how to deliver secure web applications. * OWASP ZAP (ZAP) is one of the world’s most popular free security tools and is actively maintained by hundreds of international volunteers. It can help to find security vulnerabilities in web applications. It’s also a great tool for experienced pen testers and beginners. * ZAP is what is known as a “man-in-the-middle proxy.” It stands between the browser and the web application. While you navigate through all the features of the website, it captures all actions. Then it attacks the website with known techniques to find security vulnerabilities. * It is one of the most active Open Web Application Security Project (OWASP) projects and has been given Flagship status. * When used as a proxy server it allows the user to manipulate all of the traffic that passes through it, including traffic using https. * It can also run in a daemon mode which is then controlled via a REST API. * ZAP was added to the ThoughtWorks Technology Radar in May 2015 in the Trial ring. * ZAP was originally forked from Paros, another pentesting proxy. Simon Bennetts, the project lead, stated in 2014 that only 20% of ZAP's source code was still from Paros.   **Steps:**   1. **Install owasp-zap by typing – sudo apt install owasp-zap**   **If it shows error try – sudo apt install owasp-zaproxy or sudo apt install zaproxy**  **If this also shows error means we need to update kali for owasp package.**  **Type – sudo apt-get update**     1. **Install zaproxy – sudo apt install zaproxy**      1. **Go to Application menu and type zap and start that application. Now it will take some time for start the main interface.**  * Spidering the web application * Spidering a web application means crawling all the links and getting the structure of the application. ZAP provides two spiders for crawling web applications; * The traditional ZAP spider discovers links by examining the HTML in responses from the web application. This spider is fast, but it is not always effective when exploring an AJAX web application. * This is more likely to be effective for AJAX applications. This spider explores the web application by invoking browsers which then follow the links that have been generated. The AJAX spider is slower than the traditional spider.      1. **Click Automated Scan – Enter url in URL section to attack: http://charusat.ac.in and then click Attack.**      1. **We can check for the alerts that are present on the website by clicking on Alerts and see the risk of that particular whether it is High, Medium, or Low.**      1. **Let the scan get complete and we can get all the vulnerabilities present in the website and all the alerts so that we can work on those to make it more secure.** | | |
| **Conclusion/Summary:** | | |
| **Student Signature & Date** | **Marks:** | **Evaluator Signature & Date** |

**Practical 11**

|  |  |  |
| --- | --- | --- |
| **Date: / /2023** | | |
| **Aim:** Implement hash function for the given Scenario. | | |
| **Solution:**  A Hash Function is a function that converts a given numeric or alphanumeric key to a small practical integer value. The mapped integer value is used as an index in the hash table. In simple terms, a hash function maps a significant number or string to a small integer that can be used as the index in the hash table.  **Code:**  #include<bits/stdc++.h>  using namespace std;  class Hash{  int BUCKET;  list<int> \*table;  public:  Hash(int V);  void insertItem(int x);  void deleteItem(int key);  int hashFunction(int x) {  return (x % BUCKET);  }  void displayHash();  };  Hash::Hash(int b){  this->BUCKET = b;  table = new list<int>[BUCKET];  }  void Hash::insertItem(int key){  int index = hashFunction(key);  table[index].push\_back(key);  }  void Hash::deleteItem(int key){  int index = hashFunction(key);  list <int> :: iterator i;  for (i = table[index].begin();  i != table[index].end(); i++) {  if (\*i == key)  break;  }  if (i != table[index].end())  table[index].erase(i);  }  void Hash::displayHash() {  for (int i = 0; i< BUCKET; i++) {  cout<<i;  for (auto x : table[i])  cout<< " --> " << x;  cout<<endl;  }  }  int main(){  int a[] = {15, 11, 27, 8, 12};  int n = sizeof(a)/sizeof(a[0]);  Hash h(7);  for (int i = 0; i< n; i++)  h.insertItem(a[i]);  h.deleteItem(12);  h.displayHash();  return 0;  }  **Output:** | | |
| **Conclusion/Summary:** | | |
| **Student Signature & Date** | **Marks:** | **Evaluator Signature & Date** |

**Practical 12**

|  |  |  |
| --- | --- | --- |
| **Date: / /2023** | | |
| **Aim:** Implement DES/AES algorithm for the given plain text. | | |
| **Solution:**  **DES:**  **Code:**  #include <bits/stdc++.h>  using namespace std;  string hex2bin(string s)  {  // hexadecimal to binary conversion  unordered\_map<char, string>mp;  mp['0'] = "0000";  mp['1'] = "0001";  mp['2'] = "0010";  mp['3'] = "0011";  mp['4'] = "0100";  mp['5'] = "0101";  mp['6'] = "0110";  mp['7'] = "0111";  mp['8'] = "1000";  mp['9'] = "1001";  mp['A'] = "1010";  mp['B'] = "1011";  mp['C'] = "1100";  mp['D'] = "1101";  mp['E'] = "1110";  mp['F'] = "1111";  string bin = "";  for (int i = 0; i<s.size(); i++) {  bin += mp[s[i]];  }  return bin;  }  string bin2hex(string s)  {  // binary to hexadecimal conversion  unordered\_map<string, string>mp;  mp["0000"] = "0";  mp["0001"] = "1";  mp["0010"] = "2";  mp["0011"] = "3";  mp["0100"] = "4";  mp["0101"] = "5";  mp["0110"] = "6";  mp["0111"] = "7";  mp["1000"] = "8";  mp["1001"] = "9";  mp["1010"] = "A";  mp["1011"] = "B";  mp["1100"] = "C";  mp["1101"] = "D";  mp["1110"] = "E";  mp["1111"] = "F";  string hex = "";  for (int i = 0; i<s.length(); i += 4) {  string ch = "";  ch += s[i];  ch += s[i + 1];  ch += s[i + 2];  ch += s[i + 3];  hex += mp[ch];  }  return hex;  }  string permute(string k, int\* arr, int n)  {  string per = "";  for (int i = 0; i< n; i++) {  per += k[arr[i] - 1];  }  return per;  }  string shift\_left(string k, int shifts)  {  string s = "";  for (int i = 0; i< shifts; i++) {  for (int j = 1; j < 28; j++) {  s += k[j];  }  s += k[0];  k = s;  s = "";  }  return k;  }  string xor\_(string a, string b)  {  string ans = "";  for (int i = 0; i<a.size(); i++) {  if (a[i] == b[i]) {  ans += "0";  }  else {  ans += "1";  }  }  return ans;  }  string encrypt(string pt, vector<string>rkb,  vector<string>rk)  {  // Hexadecimal to binary  pt = hex2bin(pt);  // Initial Permutation Table  int initial\_perm[64]  = { 58, 50, 42, 34, 26, 18, 10, 2, 60, 52, 44,  36, 28, 20, 12, 4, 62, 54, 46, 38, 30, 22,  14, 6, 64, 56, 48, 40, 32, 24, 16, 8, 57,  49, 41, 33, 25, 17, 9, 1, 59, 51, 43, 35,  27, 19, 11, 3, 61, 53, 45, 37, 29, 21, 13,  5, 63, 55, 47, 39, 31, 23, 15, 7 };  // Initial Permutation  pt = permute(pt, initial\_perm, 64);  cout<< "After initial permutation: " << bin2hex(pt)  <<endl;  // Splitting  string left = pt.substr(0, 32);  string right = pt.substr(32, 32);  cout<< "After splitting: L0=" << bin2hex(left)  << " R0=" << bin2hex(right) <<endl;  // Expansion D-box Table  int exp\_d[48]  = { 32, 1, 2, 3, 4, 5, 4, 5, 6, 7, 8, 9,  8, 9, 10, 11, 12, 13, 12, 13, 14, 15, 16, 17,  16, 17, 18, 19, 20, 21, 20, 21, 22, 23, 24, 25,  24, 25, 26, 27, 28, 29, 28, 29, 30, 31, 32, 1 };  // S-box Table  int s[8][4][16] = {  { 14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5,9, 0, 7, 0, 15, 7, 4, 14, 2, 13, 1, 10, 6,  12, 11, 9, 5, 3, 8, 4, 1, 14, 8, 13, 6, 2,11, 15, 12, 9, 7, 3, 10, 5, 0, 15, 12, 8, 2,  4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13 },  { 15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12,0, 5, 10, 3, 13, 4, 7, 15, 2, 8, 14, 12, 0,  1, 10, 6, 9, 11, 5, 0, 14, 7, 11, 10, 4, 13,1, 5, 8, 12, 6, 9, 3, 2, 15, 13, 8, 10, 1,  3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14, 9 },  { 10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12,7, 11, 4, 2, 8, 13, 7, 0, 9, 3, 4,  6, 10, 2, 8, 5, 14, 12, 11, 15, 1, 13,6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12,  5, 10, 14, 7, 1, 10, 13, 0, 6, 9, 8,7, 4, 15, 14, 3, 11, 5, 2, 12 },  { 7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11,12, 4, 15, 13, 8, 11, 5, 6, 15, 0, 3, 4, 7,  2, 12, 1, 10, 14, 9, 10, 6, 9, 0, 12, 11, 7,13, 15, 1, 3, 14, 5, 2, 8, 4, 3, 15, 0, 6,  10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14 },  { 2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13,0, 14, 9, 14, 11, 2, 12, 4, 7, 13, 1, 5, 0,  15, 10, 3, 9, 8, 6, 4, 2, 1, 11, 10, 13, 7,8, 15, 9, 12, 5, 6, 3, 0, 14, 11, 8, 12, 7,  1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3 },  { 12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14,7, 5, 11, 10, 15, 4, 2, 7, 12, 9, 5, 6, 1,  13, 14, 0, 11, 3, 8, 9, 14, 15, 5, 2, 8, 12,3, 7, 0, 4, 10, 1, 13, 11, 6, 4, 3, 2, 12,  9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13 },  { 4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5,10, 6, 1, 13, 0, 11, 7, 4, 9, 1, 10, 14, 3,  5, 12, 2, 15, 8, 6, 1, 4, 11, 13, 12, 3, 7,14, 10, 15, 6, 8, 0, 5, 9, 2, 6, 11, 13, 8,  1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12 },  { 13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5,0, 12, 7, 1, 15, 13, 8, 10, 3, 7, 4, 12, 5,  6, 11, 0, 14, 9, 2, 7, 11, 4, 1, 9, 12, 14,2, 0, 6, 10, 13, 15, 3, 5, 8, 2, 1, 14, 7,  4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11 }  };  int per[32]  = { 16, 7, 20, 21, 29, 12, 28, 17, 1, 15, 23,  26, 5, 18, 31, 10, 2, 8, 24, 14, 32, 27,  3, 9, 19, 13, 30, 6, 22, 11, 4, 25 };  cout<<endl;  for (int i = 0; i< 16; i++) {  string right\_expanded = permute(right, exp\_d, 48);  string x = xor\_(rkb[i], right\_expanded);  string op = "";  for (int i = 0; i< 8; i++) {  int row = 2 \* int(x[i \* 6] - '0')  + int(x[i \* 6 + 5] - '0');  int col = 8 \* int(x[i \* 6 + 1] - '0')  + 4 \* int(x[i \* 6 + 2] - '0')  + 2 \* int(x[i \* 6 + 3] - '0')  + int(x[i \* 6 + 4] - '0');  int val = s[i][row][col];  op += char(val / 8 + '0');  val = val % 8;  op += char(val / 4 + '0');  val = val % 4;  op += char(val / 2 + '0');  val = val % 2;  op += char(val + '0');  }  op = permute(op, per, 32);  x = xor\_(op, left);  left = x;  if (i != 15) {  swap(left, right);  }  cout<< "Round " <<i + 1 << " " << bin2hex(left)  << " " << bin2hex(right) << " " <<rk[i]  <<endl;  }  string combine = left + right;  int final\_perm[64]  = { 40, 8, 48, 16, 56, 24, 64, 32, 39, 7, 47,15, 55, 23, 63, 31, 38, 6, 46, 14, 54, 22,  62, 30, 37, 5, 45, 13, 53, 21, 61, 29, 36,4, 44, 12, 52, 20, 60, 28, 35, 3, 43, 11,  51, 19, 59, 27, 34, 2, 42, 10, 50, 18, 58,26, 33, 1, 41, 9, 49, 17, 57, 25 };  string cipher  = bin2hex(permute(combine, final\_perm, 64));  return cipher;  }  int main()  {  string pt, key;  /\*cout<<"Enter plain text(in hexadecimal): ";  cin>>pt;  cout<<"Enter key(in hexadecimal): ";  cin>>key;\*/  pt = "123456ABCD132536";  key = "AABB09182736CCDD";  key = hex2bin(key);  int keyp[56]  = { 57, 49, 41, 33, 25, 17, 9, 1, 58, 50, 42, 34,26, 18, 10, 2, 59, 51, 43, 35, 27, 19, 11, 3,60, 52, 44, 36, 63, 55, 47, 39, 31, 23, 15, 7,62, 54, 46, 38, 30, 22, 14, 6, 61, 53, 45, 37,29, 21, 13, 5, 28, 20, 12, 4 };  key = permute(key, keyp, 56);  int shift\_table[16] = { 1, 1, 2, 2, 2, 2, 2, 2,  1, 2, 2, 2, 2, 2, 2, 1 };  int key\_comp[48] = { 14, 17, 11, 24, 1, 5, 3, 28,15, 6, 21, 10, 23, 19, 12, 4,26, 8, 16, 7, 27, 20, 13, 2,41, 52, 31, 37, 47, 55, 30, 40,51, 45, 33, 48, 44, 49, 39, 56,34, 53, 46, 42, 50, 36, 29, 32 };  string left = key.substr(0, 28);  string right = key.substr(28, 28);  vector<string>rkb; // rkb for RoundKeys in binary  vector<string>rk; // rk for RoundKeys in hexadecimal  for (int i = 0; i< 16; i++) {  // Shifting  left = shift\_left(left, shift\_table[i]);  right = shift\_left(right, shift\_table[i]);  string combine = left + right;  string RoundKey = permute(combine, key\_comp, 48);  rkb.push\_back(RoundKey);  rk.push\_back(bin2hex(RoundKey));  }  cout<< "\nEncryption:\n\n";  string cipher = encrypt(pt, rkb, rk);  cout<< "\nCipher Text: " << cipher <<endl;  cout<< "\nDecryption\n\n";  reverse(rkb.begin(), rkb.end());  reverse(rk.begin(), rk.end());  string text = encrypt(cipher, rkb, rk);  cout<< "\nPlain Text: " << text <<endl;  }  **Encryption:**    **Decryption:**    **AES:**  **Code:**  #include <stdio.h>  #include <iostream>  #include <stdlib.h>  #include <string.h>  using namespace std;  #define Nb 4  int Nr=0;  int Nk=0;.  unsigned char in[1024], out[1024], state[4][Nb];  unsigned char RoundKey[240];  unsigned char Key[32];  int getSBoxValue(int num) {  int sbox[256] = {  0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5,  0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76, //0  0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0,  0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0, //1  0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc,  0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15, //2  0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a,  0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75, //3  0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0,  0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84, //4  0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b,  0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf, //5  0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85,  0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8, //6  0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5,  0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2, //7  0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17,  0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73, //8  0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88,  0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb, //9  0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c,  0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79, //A  0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9,  0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08, //B  0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6,  0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a, //C  0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e,  0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e, //D  0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94,  0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf, //E  0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68,  0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16 }; //F  return sbox[num];}  int Rcon[255] = {  0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20,  0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8, 0xab, 0x4d,  0x9a, 0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35,  0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef, 0xc5, 0x91,  0x39, 0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f,  0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc, 0x83, 0x1d,  0x3a, 0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04,  0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36, 0x6c,  0xd8, 0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63,  0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3, 0x7d, 0xfa,  0xef, 0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd,  0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94, 0x33, 0x66,  0xcc, 0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d,  0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80,  0x1b, 0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a, 0x2f,  0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35, 0x6a, 0xd4,  0xb3, 0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39, 0x72,  0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f, 0x25, 0x4a,  0x94, 0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a, 0x74,  0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04, 0x08, 0x10,  0x20, 0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8, 0xab,  0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97,  0x35, 0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef, 0xc5,  0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2,  0x9f, 0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc, 0x83,  0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02,  0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36,  0x6c, 0xd8, 0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc,  0x63, 0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3, 0x7d,  0xfa, 0xef, 0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3,  0xbd, 0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94, 0x33,  0x66, 0xcc, 0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb };  void KeyExpansion() {  int i,j;  unsigned char temp[4],k;  for (i=0 ; i<Nk ; i++) {  RoundKey[i\*4] = Key[i\*4];  RoundKey[i\*4+1] = Key[i\*4+1];  RoundKey[i\*4+2] = Key[i\*4+2];  RoundKey[i\*4+3] = Key[i\*4+3];  }  while (i< (Nb \* (Nr+1))) {  for (j=0 ; j < 4 ; j++) {  temp[j] = RoundKey[(i-1) \* 4 + j];  }  if (i % Nk == 0) {  k = temp[0];  temp[0] = temp[1];  temp[1] = temp[2];  temp[2] = temp[3];  temp[3] = k;  temp[0] = getSBoxValue(temp[0]);  temp[1] = getSBoxValue(temp[1]);  temp[2] = getSBoxValue(temp[2]);  temp[3] = getSBoxValue(temp[3]);  temp[0] = temp[0] ^ Rcon[i/Nk];  }  else if (Nk> 6 &&i % Nk == 4) {  temp[0] = getSBoxValue(temp[0]);  temp[1] = getSBoxValue(temp[1]);  temp[2] = getSBoxValue(temp[2]);  temp[3] = getSBoxValue(temp[3]);  }  RoundKey[i\*4+0] = RoundKey[(i-Nk)\*4+0] ^ temp[0];  RoundKey[i\*4+1] = RoundKey[(i-Nk)\*4+1] ^ temp[1];  RoundKey[i\*4+2] = RoundKey[(i-Nk)\*4+2] ^ temp[2];  RoundKey[i\*4+3] = RoundKey[(i-Nk)\*4+3] ^ temp[3];  i++;  }}  void AddRoundKey(int round) {  int i,j;  for (i=0 ; i< Nb ; i++) {  for (j=0 ; j < 4 ; j++) {  state[j][i] ^= RoundKey[round \* Nb \* 4 + i \* Nb + j];  }  }  }  void SubBytes() {  int i,j;  for (i=0 ; i< 4 ; i++) {  for (j=0 ; j < Nb ; j++) {  state[i][j] = getSBoxValue(state[i][j]);  }  }  }  void ShiftRows() {  unsigned char temp;  temp = state[1][0];  state[1][0] = state[1][1];  state[1][1] = state[1][2];  state[1][2] = state[1][3];  state[1][3] = temp;  temp = state[2][0];  state[2][0] = state[2][2];  state[2][2] = temp;  temp = state[2][1];  state[2][1] = state[2][3];  state[2][3] = temp;  temp = state[3][0];  state[3][0] = state[3][3];  state[3][3] = state[3][2];  state[3][2] = state[3][1];  state[3][1] = temp;  }  void MixColumns() {  int i;  unsigned char Tmp,Tm,t;  for (i=0 ; i< Nb ; i++) {  t = state[0][i];  Tmp = state[0][i] ^ state[1][i] ^ state[2][i] ^ state[3][i] ;  Tm = state[0][i] ^ state[1][i] ;  Tm = xtime(Tm);  state[0][i] ^= Tm ^ Tmp ;  Tm = state[1][i] ^ state[2][i] ;  Tm = xtime(Tm);  state[1][i] ^= Tm ^ Tmp ;  Tm = state[2][i] ^ state[3][i] ;  Tm = xtime(Tm);  state[2][i] ^= Tm ^ Tmp ;  Tm = state[3][i] ^ t ;  Tm = xtime(Tm);  state[3][i] ^= Tm ^ Tmp ;  }  }  void Cipher() {  int i,j,round=0;  for (i=0 ; i< Nb ; i++) {  for (j=0 ; j < 4 ; j++) {  state[j][i] = in[i\*4 + j]; }  }  AddRoundKey(0);  for (round=1 ; round < Nr ; round++) {  SubBytes();  ShiftRows();  MixColumns();  AddRoundKey(round);  }  SubBytes();  ShiftRows();  AddRoundKey(Nr);  for (i=0 ; i< Nb ; i++) {  for (j=0 ; j < 4 ; j++) {  out[i\*4+j]=state[j][i];  }}}  int fillBlock (int sz, char \*str, unsigned char \*in) {  int j=0;  while (sz<strlen(str)) {  if (j >= Nb\*4) break;  in[j++] = (unsigned char)str[sz];  sz++;}  // Pad the block with 0s, if necessary  if (sz>= strlen(str)) for ( ; j < Nb\*4 ; j++) in[j] = 0;  return sz;  }  int main(int argc, char \*\*argv) {  int i;  if (argc != 2) {  cerr<< "Usage: " <<argv[0] << " <keysize: 1=128, 2=192, 3=256>\n";  exit(0); }  switch (atoi(argv[1])) {  case 1: Nk = 4; break;  case 2: Nk = 6; break;  case 3: Nk = 8; break;  default: Nk = 4; break; }  Nr = Nk + 6;  Key[0] = 0x2b; Key[1] = 0x7e; Key[2] = 0x15; Key[3] = 0x16;  Key[4] = 0x28; Key[5] = 0xae; Key[6] = 0xd2; Key[7] = 0xa6;  Key[8] = 0xab; Key[9] = 0xf7; Key[10] = 0x15; Key[11] = 0x88;  Key[12] = 0x09; Key[13] = 0xcf; Key[14] = 0x4f; Key[15] = 0x3c;  char str[1024];  fgets(str, 1024, stdin);  KeyExpansion();  int sz=0;  while (sz<strlen(str)) {  sz = fillBlock (sz, str, in);  Cipher();  for (i=0 ; i< Nb\*4 ; i++) cout<< (int)out[i] << " ";  }  printf("\n\n");  }  **Output:** | | |
| **Conclusion/Summary:** | | |
| **Student Signature & Date** | **Marks:** | **Evaluator Signature & Date** |