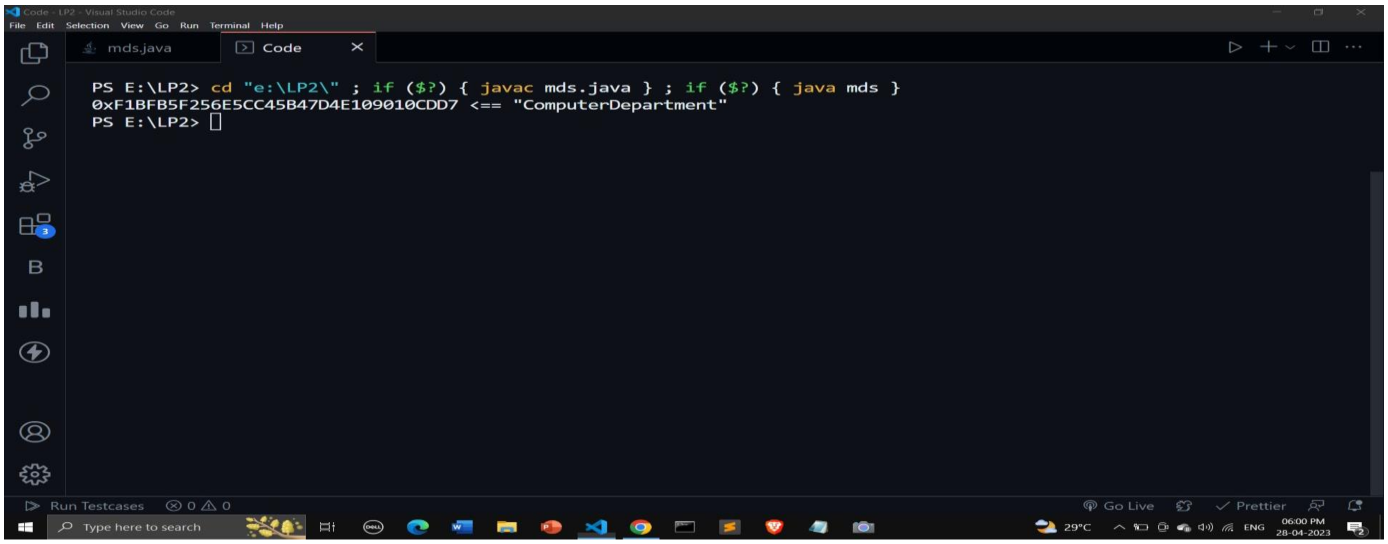
MD5 algorithm

```
public class MD5 {
  private static final int INIT_A = 0x67452301;
  private static final int INIT_B = (int) 0xEFCDAB89 L;
  private static final int INIT_C = (int) 0x98BADCFE L;
  private static final int INIT_D = 0x10325476;
  private static final int[] SHIFT_AMTS = {
    7,    12,    17,    22,    5,    9,    14,    20,    4,    11,    16,    23,    6,    10,    15,    21
  };
  private static final int[] TABLE_T = new int[64];
  static {
    for (int i = 0; i < 64; i++)
      TABLE_T[i] = (int)(long)((1 L << 32) * Math.abs(Math.sin(i + 1)));
  }
  public static byte[] computeMD5(byte[] message) {
    int messageLenBytes = message.length;
    int numBlocks = ((messageLenBytes + 8) >>> 6) + 1;
    int totalLen = numBlocks << 6;
    byte[] paddingBytes = new byte[totalLen - messageLenBytes];
    paddingBytes[0] = (byte) 0x80;
    long messageLenBits = (long) messageLenBytes << 3;
    for (int i = 0; i < 8; i++) {
      paddingBytes[paddingBytes.length - 8 + i] = (byte) messageLenBits;
      messageLenBits >>>= 8;
    }
    int a = INIT_A;
    int b = INIT_B;
    int c = INIT_C;
    int d = INIT_D;
    int[] buffer = new int[16];
    for (int i = 0; i < numBlocks; i++) {
      int index = i << 6;
      for (int j = 0; j < 64; j++, index++)
        buffer[j >>> 2] = ((int)((index < messageLenBytes) ? message[index] :
          paddingBytes[index - messageLenBytes]) << 24) | (buffer[j >>> 2] >>> 8);
      int originalA = a;
      int originalB = b;
      int originalC = c;
      int originalD = d;
      for (int j = 0; j < 64; j++) {
        int div16 = j >>> 4;
        int f = 0;
        int bufferIndex = j;
        switch (div16) {
        case 0:
          f = (b & c) | (~b & d);
          break;
        case 1:
          f = (b & d) | (c & ~d);
          bufferIndex = (bufferIndex * 5 + 1) & 0x0F;
          break;
        case 2:
          f = b ^ c ^ d;
          bufferIndex = (bufferIndex * 3 + 5) & 0x0F;
          break;
        case 3:
          f = c ^ (b | ~d);
          bufferIndex = (bufferIndex * 7) & 0x0F;
          break;
        }
        int temp = b + Integer.rotateLeft(a + f + buffer[bufferIndex] + TABLE_T[j],
          SHIFT_AMTS[(div16 << 2) | (j & 3)]);
        a = d;
        d = c;
        c = b;
        b = temp;
      }
      a += originalA;
      b += originalB;
      c += originalC;
      d += originalD;
    }
    byte[] md5 = new byte[16];
    int count = 0;
    for (int i = 0; i < 4; i++) {
      int n = (i == 0) ? a : ((i == 1) ? b : ((i == 2) ? c : d));
      for (int j = 0; j < 4; j++) {
        md5[count++] = (byte) n;
        n >>>= 8;
      }
    }
    return md5;
  }
  public static String toHexString(byte[] b) {
    StringBuilder sb = new StringBuilder();
    for (int i = 0; i < b.length; i++) {
      sb.append(String.format("%02X", b[i] & 0xFF));
    }
    return sb.toString();
  }
  public static void main(String[] args) {
      String[] testStrings = {
        "ComputerDepartment"
      };
      for (String s: testStrings)
        System.out.println("0x" + toHexString(computeMD5(s.getBytes())) + " <== \"" + s + "\"");
      return; }
}
```

Output:



```
PS E:\LP2> cd "e:\LP2\" ; if ($?) { javac mds.java } ; if ($?) { java mds }
0xF1BFB5F256E5CC45B47D4E109010CDD7 <== "ComputerDepartment"
PS E:\LP2> 
```