# Joseph Morgan

**Homework 10**

**CISP440**

## 1 Source code for inverse table generator program

### inverse_table.h

```
#ifndef _INVERSE_TABLE_H_
#define _INVERSE_TABLE_H_

static unsigned char fi[4][256];

void fill_fi();
#endif
```

### inverse_table.cpp

```
#include "encryptor.h"
#include "inverse_table.h"

void fill_fi()
{
  unsigned char temp;
  for (int i = 0; i < 4; ++i) {
    for (int j = 0; j < 256; ++j) {
      temp = f[i][j];
      fi[i][temp] = j;
    }
  }
}
```

## 2 Source code for decryption program

### decrytor.h

```
#ifndef _DECRYPTOR_H_
#define _DECRYPTOR_H_

#include <fstream>

unsigned char decrypt (unsigned char w, unsigned char key);
void decrypt_file (std::ifstream &infile, std::ofstream &outfile, unsigned
    char key);
void decrypt_without_key (std::ifstream &infile, std::ofstream &outfile);

#endif
```

### decrytor.cpp

```
#include <climits>
#include <iostream>
#include "inverse_table.h"
#include "decryptor.h"
#include "encryptor.h"

```

```
unsigned char decrypt (unsigned char w, unsigned char key)
{
  unsigned char o_char = 0;
  unsigned char x0, y0, z0 = 0;
  unsigned char x1, y1, z1 = 0;
  unsigned char x2, y2, z2 = 0;
  unsigned char x3, y3 = 0;
  unsigned char p, q, r, s = 0;

  p = (key & 0x03);
  q = (key & 0x0C) >> 2;
  r = (key & 0x30) >> 4;
  s = (key & 0xC0) >> 6;

  y3 = w ^ key;
  x3 = swapbytes (y3);
  z2 = fi[p][x3];

  y2 = z2 ^ key;
  x2 = swapbytes (y2);
  z1 = fi[q][x2];

  y1 = z1 ^ key;
  x1 = swapbytes (y1);
  z0 = fi[r][x1];

  y0 = z0 ^ key;
  x0 = swapbytes (y0);
  o_char = fi[s][x0];

  return o_char;
}

void decrypt_file (std::ifstream &infile, std::ofstream &outfile, unsigned
    char key)
{
  char c;
  while (!infile.eof())
  {
    infile.read(&c, 1);
    if (!infile.eof())
    {
      c = decrypt(c, key);
      outfile.write(&c, 1);
    }
  }
}

void decrypt_without_key (std::ifstream &infile, std::ofstream &outfile)
{
  short key;
  char *c = (char*)malloc(2 * sizeof(char));
  char *d = (char*)malloc(2 * sizeof(char));
  infile.read (c, 2);
  for (key = 0; key <= UCHAR_MAX; ++key) {
    d[0] = decrypt (c[0], key);
    d[1] = decrypt (c[1], key);
    if (d[0] == 'B' && d[1] == 'M') {
      std::cout << "Key found as: " << (char)key << std::endl;
      break;
    }
  }
  if (key > UCHAR_MAX) {
    std::cout << "Key could not be found\n";
    exit(1);
  } else {
    infile.seekg(0, infile.beg);
    decrypt_file (infile, outfile, key);
  }
}
```

## main.cpp

```
#include <string.h>
#include <iostream>
#include "inverse_table.h"
#include "encryptor.h"
#include "decryptor.h"

int main (int argc, char* argv[])
{
  fill_fi();
  char selection;
  std::ifstream infile;
  std::ofstream outfile;
  std::string filename;
  char key = 0;

  if (argc >= 3 && strcmp(argv[1], "-f") == 0) {
    filename = argv[2];
    infile.open(filename, std::ios_base::binary);
    if (!infile)
      std::cout << "Invalid filename!\n", exit(1);
    outfile.open("cracked.bmp", std::ios_base::binary);
    decrypt_without_key (infile, outfile);
    if (argc == 4 && strcmp(argv[3], "-o") == 0) {
      for (int i = 0; i < 4; ++i) {
        std::cout << "{";
        for (int j = 0; j < 256; ++j) {
          std::cout << " " << fi[i][j];
        }
        std::cout << " }" << std::endl;
      }
    }
  } else {

    std::cout
      << "*** miniDES encryption ***\n"
      << "  e) Encrypt file\n"
      << "  d) Decrypt file\n"
      << "  u) Decrypt file with unknown key"
      << "\nPlease make a selection: ";
    std::cin >> selection;
    std::cout << std::endl;

    switch (selection) {
      case ('e') :
        std::cout
          << "You've selected 'encrypt file'\n"
          << "Please input the filename: ";
        std::cin >> filename;
        std::cout << std::endl << "Please input the encryption key: ";
        std::cin >> key;
        std::cout << std::endl;
        infile.open (filename, std::ios_base::binary);
        if (!infile)
          std::cerr << "Input could not be opened\n";
        outfile.open ("encrypted.bmp", std::ios_base::binary);
        encrypt_file (infile, outfile, key);
        infile.close();
        outfile.close();
        break;

      case ('d') :
        std::cout
          << "You've selected 'decrypt file'\n"
          << "Please input the filename: ";
        std::cin >> filename;
        std::cout << std::endl << "Please input the encryption key: ";
        std::cin >> key;
        std::cout << std::endl;
        infile.open (filename, std::ios_base::binary);
        if (!infile)
          std::cerr << "Input could not be opened\n";
        outfile.open ("decrypted.bmp", std::ios_base::binary);
        decrypt_file (infile, outfile, key);
        infile.close();
        outfile.close();
        break;

      case ('u') :
        std::cout
          << "You've selected 'decrypt file with unknown key'\n"
          << "Please input the filename: ";
        std::cin >> filename;
        infile.open (filename, std::ios_base::binary);
        if (!infile)
          std::cerr << "Input could not be opened\n";
        outfile.open ("cracked.bmp", std::ios_base::binary);
        decrypt_without_key (infile, outfile);
        infile.close();
        outfile.close();
```

```
          break;

        default:
          std::cout << "Inavlid entry\n", exit(1);
      }
    }
}
```

## 3 Inverse function tables printout

## 4 Key value

## 5 Inverse function composition order

## 6 Decrypted Output