Joseph Morgan

Homework 10

CISP440

1 Source code for inverse table generator program

inverse_table.h

```
#ifndef _INVERSE_TABLE_H_
#define _INVERSE_TABLE_H_

extern unsigned char fi[4][256];

void fill_fi();
#endif
```

inverse_table.cpp

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

unsigned char fi[4][256];

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 << " " << (int)fi[i][j];
        }
        std::cout << " }" << std::endl;
      }
    }
  } else {

```

```cpp
    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

```
{ 30 0 124 209 229 29 15 154 68 23 28 196 22 235 250 86 27 238 1 93 249 187
   165 132 128 112 230 231 50 26 232 31 3 109 223 169 94 185 46 102 103 25
   208 121 135 239 246 90 38 139 241 41 179 51 83 67 234 64 81 20 87 164 108
   44 21 149 82 148 219 133 253 98 57 80 36 140 32 186 212 251 160 24 88 221
   247 77 85 71 198 248 65 244 166 236 252 104 218 225 153 58 101 105 195 183
    66 37 226 163 233 56 220 91 19 254 245 33 202 255 242 184 156 2 126 45 39
    222 206 4 224 188 129 69 6 78 203 141 7 5 210 106 237 84 89 167 8 159 150
    155 59 16 115 189 9 197 211 107 228 92 60 10 162 130 207 95 199 34 96 11
   35 70 190 131 40 240 137 12 13 110 14 17 72 134 42 125 97 170 18 79 243 73
    136 191 181 168 76 43 171 182 192 99 47 100 111 157 172 48 193 158 142
   194 49 216 52 74 75 53 113 54 213 138 55 161 61 114 173 143 200 116 201 62
    63 174 117 204 144 118 127 119 120 152 122 123 145 146 147 151 175 214
   176 177 178 180 215 205 217 227 }
{ 100 24 48 128 55 204 205 191 122 70 218 107 61 228 138 251 240 232 106 215
   37 41 68 178 152 8 67 26 108 166 89 77 183 84 222 79 65 75 123 57 137 158
   69 212 9 109 43 102 187 90 150 206 220 64 71 201 241 35 133 159 6 101 124
   219 62 54 221 116 135 85 197 23 173 246 239 189 148 249 254 129 190 103 51
    172 19 25 78 27 226 179 170 184 36 223 176 110 255 167 111 28 146 83 119
   1 242 56 33 80 104 244 202 238 0 245 139 160 86 21 30 72 117 157 252 140
   74 115 194 162 3 141 224 99 233 171 88 161 87 73 38 180 4 195 91 92 142
   237 229 203 39 163 130 253 177 207 192 76 5 149 125 45 93 210 112 126 20
   199 49 151 234 14 63 13 250 208 113 114 98 58 136 94 11 40 66 227 164 42
   81 12 214 168 165 181 247 95 118 143 155 15 200 147 182 120 185 144 209
   131 211 213 153 188 193 121 154 10 32 169 82 156 198 7 96 17 248 216 2 22
   44 186 16 231 230 59 196 105 217 97 18 127 132 34 46 134 225 145 174 235
   236 243 175 29 31 47 50 52 53 60 }
{ 27 79 250 89 136 197 166 103 99 98 127 14 30 102 19 176 96 108 226 221 74
   157 194 35 254 36 130 40 100 225 206 247 33 101 82 152 243 15 37 227 31
   164 90 149 62 229 2 165 65 141 52 208 181 135 128 123 70 97 143 186 131
   253 7 172 224 137 175 11 1 178 212 0 53 21 235 60 66 246 119 68 104 54 45
   12 138 125 86 200 237 191 118 255 203 223 43 228 105 173 87 218 57 22 129
   106 177 156 121 41 179 3 59 56 196 77 23 185 71 233 16 180 252 42 126 32
   198 240 4 61 170 182 147 46 38 163 244 192 148 34 78 132 24 115 25 39 117
   190 55 5 133 50 110 230 122 134 241 64 248 153 83 107 199 139 171 47 72
   195 245 183 238 140 184 120 249 174 209 187 124 111 216 6 188 193 63 8 112
    189 20 142 231 67 232 13 201 17 234 9 48 217 73 236 202 204 239 44 219 28
    49 151 242 75 69 251 113 10 76 205 18 207 51 58 80 81 144 210 145 84 146
   214 109 85 211 26 29 213 88 215 91 92 93 114 169 94 220 95 116 150 154 167
    155 158 159 160 161 162 168 222 }
{ 178 169 242 58 238 90 235 105 80 167 236 248 51 145 186 32 59 53 215 243 131
    88 188 147 247 68 99 162 254 211 9 249 93 201 50 157 89 102 237 239 25 60
```

91 34 86 63 104 244 47 27 227 148 85 202 175 232 113 101 78 246 196 61
168 152 10 33 12 116 165 7 114 45 250 212 3 219 16 107 220 184 137 13 14
28 176 228 231 171 229 221 120 204 67 181 81 94 132 87 135 170 30 100 1
141 179 20 240 11 172 62 153 216 18 180 92 117 217 206 241 66 26 82 234 15
197 106 136 182 143 21 173 31 17 166 35 118 19 174 39 149 103 222 44 189
146 183 218 133 29 177 76 23 213 108 185 159 144 129 223 255 245 2 130 194
187 251 95 83 224 52 190 109 150 119 36 203 209 191 138 54 192 193 139
151 230 210 195 110 252 37 200 198 126 199 22 121 96 122 64 134 225 97 154
111 112 46 253 115 155 123 124 125 6 4 41 24 38 48 57 98 40 127 226 42 0
43 128 49 140 142 69 156 233 214 5 158 79 55 56 65 205 160 207 70 71 208 8
72 73 74 75 84 161 77 163 164 }

4 Key value

Key value was: c

5 Inverse function composition order

I know you said this part was easier than it sounds, but I'm really at a loss for what it means and at this point I've waited too long to email and ask.. My loss I guess :(I feel like the answer to this has got to be in my code somewhere but I can't seem to work out what exactly I should put in this section.

6 Decrypted Output