CS205 C/C++ Programming Lab Assignment 4

Name: 钟兆玮 (Zhaowei Zhong)
SID: 11611722

Part 1 - Analysis

Download (with wget) http://www.unicode.org/Public/8.0.0/ucd/Blocks.txt
1) Define a suitable structure to load all this in an array (size 300 is big enough)
2) Write a function to search this array when provided with a Unicode value, and a small test program.
You are provided with code that does Unicode/UTF-8 conversions
Read a file from the standard input - that means that your program will be called like this: ./your_program < name_of_file_to_analyze
Your program must display on the standard output the name of the block to which most characters belong (there may be characters from different blocks)
We do not have a strict requirement for your output.
But the name of the block must be correct.

Notice:
In this assignment, we will judge your program by 6 text files with Unicode characters. All of them are provided in the zip file.
Answers: Armenian, Georgian, Lao, Malayalam Devanagari, Georgian (-5 for each wrong result)
Your program should analyze the text file and print the correct result instead of printing the answer directly. You will get 0 if your program prints answer directly.
You can learn how to use given utf8 functions in using_utf8_to_codepoint.c.
To help you understand how UTF-8 works, you can read https://en.wikipedia.org/wiki/UTF-8.

Part 2 - Code

```
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <math.h>
#include <algorithm>
#include <iterator>
#include <valarray>
```

```cpp
#include "utf8.h"

struct Block {
  long start;
  long end;
  std::string language;
};

std::vector<std::string> split(std::string str, std::string pattern);
Block line2block(std::string line, int line_num);
long hex2int(const std::string& hexStr);
int char2code(char* cha);
int findCategory(char* cha, Block blocks[]);

int main() {
  Block blocks[262];
  // Load Blocks
  std::ifstream infile;
  std::string line;
  infile.open("Blocks.txt");
  if (infile.fail()) {
    std::cout << "Data file is missing." << std::endl;
    exit(-1);
  }
  int i = 0;
  int line_num = 0;
  while (!infile.eof()) {
    getline(infile, line);
    line_num++;
    if (line_num < 35 || line_num > 296) {
      continue;
    }
    blocks[i] = line2block(line, i);
    i++;
  }
  // Load Test Data
  std::string in;
  std::string text;
  while (getline(std::cin, in)) {
    text = text + in;
  }
  int counts[262];
  for (int j = 0; j < 262; j++) {
    counts[j] = 0;
```

```
  }
  for (int j = 0; j < text.length(); j++) {
    char* cha = &text[j];
    int category = findCategory(cha, blocks);
    counts[category]++;
  }
  counts[1] = 0; // Ignore Latin-1 Supplement
  int result = std::distance(counts, std::max_element(counts, counts +
sizeof(counts) / sizeof(counts[0])));
  std::cout << blocks[result].language << std::endl;
}

std::vector<std::string> split(std::string str, std::string pattern) {
  std::string::size_type pos;
  std::vector<std::string> result;
  str += pattern;
  int size = str.size();
  for (int i = 0; i < size; i++) {
    pos = str.find(pattern, i);
    if (pos < size) {
      std::string s = str.substr(i, pos - i);
      result.push_back(s);
      i = pos + pattern.size() - 1;
    }
  }
  return result;
}

Block line2block(std::string line, int i) {
  if (i < 160) {
    return Block {
      hex2int(line.substr(0, 4)),
      hex2int(line.substr(6, 4)),
      line.substr(12)
    };
  } else if (i < 260) {
    return Block {
      hex2int(line.substr(0, 5)),
      hex2int(line.substr(7, 5)),
      line.substr(14)
    };
  } else {
    return Block {
      hex2int(line.substr(0, 6)),
```

```
      hex2int(line.substr(8, 6)),
      line.substr(16)
    };
  }
}

long hex2int(const std::string& hexStr) {
  char *offset;
  if (hexStr.length() > 2) {
    if (hexStr[0] == '0' && hexStr[1] == 'x') {
      return strtol(hexStr.c_str(), &offset, 0);
    }
  }
  return strtol(hexStr.c_str(), &offset, 16);
}

int char2code(char* cha) {
  unsigned char *p;
  p = (unsigned char *)cha;
  int bytes_in_char;
  unsigned int result;
  return utf8_to_codepoint(p, &bytes_in_char);
}

int findCategory(char* cha, Block blocks[]) {
  for (int i = 0; i < 262; i++) {
    int code = char2code(cha);
    if (code >= blocks[i].start && code <= blocks[i].end) {
      return i;
    } else if (code > blocks[i].end) {
      continue;
    } else {
      exit(-1);
    }
  }
  return -1;
}
```

Part 3 - Result & Verification

Test Case #1: Sample.txt

```
~/Courses/CPP/assignment4  master  ./main < test/sample.txt
Armenian
```

Test Case #2: Sample2.txt

```
~/Courses/CPP/assignment4  master  ./main < test/sample2.txt
Georgian
```

Test Case #3: Sample3.txt

```
~/Courses/CPP/assignment4  master  ./main < test/sample3.txt
Lao
```

Test Case #4: Sample4.txt

```
~/Courses/CPP/assignment4  master  ./main < test/sample4.txt
Malayalam
```

Test Case #5: Sample5.txt

```
~/Courses/CPP/assignment4  master  ./main < test/sample5.txt
Devanagari
```

Test Case #6: Sample6.txt

```
~/Courses/CPP/assignment4  master  ./main < test/sample6.txt
Georgian
```

Test Case #7: Chinese Hanzi

```
~/Courses/CPP/assignment4  master  ./main < test/sample7.txt
CJK Unified Ideographs
```

Test Case #8: Japanese Hiragana

```
~/Courses/CPP/assignment4  master  ./main < test/sample7.txt
Hiragana
```

Test Case #8: Japanese Katakana

```
~/Courses/CPP/assignment4  master  ./main < test/sample7.txt
Katakana
```

Part 4 - Difficulties & Solutions

Latin-1 Supplement block contains punctuations, so it should be ignored, otherwise it might be counted and the result of that block might be larger than the main characters.