CS205 C/ C++ Programming - Lab Assignment 4

Name: 邱煜（Qiu Yu）

SID: 11611127

Part1 - Analysis

This program is to load the codepoint data from a file `Blocks.txt` to a structure, then use that to determine the input stream's name of the block to which most its characters belongs.

Part2 - Code

```
1  //
2  //  main.cpp
3  //  lab4
4  //
5  //  Created by 邱煜 on 2019/4/12.
6  //  Copyright © 2019 邱煜. All rights reserved.
7  //
8
9  #include <cstdlib>
10 #include <iostream>
11 #include <sstream>
12 #include <fstream>
13 #include <map>
14 #include "main.hpp"
15
16 using namespace std;
17
18 int cnt = 0;
19 int maxOccurence = 0;
20 Blocks blocks[300];
21 map<int, int> occurenceMap;
22
23 int main(int argc, char **argv){
24     ifstream file ("Blocks.txt");
25     // invalid case
26     if(!file.is_open()){
27         cout << "Open file blocks.txt failed!" << endl;
28         exit(9);
```

```
    }
    // useless information
    string value;
    for (int i = 0; i < 34; i++) {
        getline(file, value);
    }
    // store data into blocks
    int numInDecimal = 0;
    stringstream ss;
    while(file.good() && cnt<262){
        getline(file, value, '.');
        if(value.length()){
            ss << value;
            ss >> hex >> numInDecimal;
            ss.clear();
            blocks[cnt].id = cnt;
            blocks[cnt].startCode = numInDecimal;
            // cout << cnt << "->" << value << "," << numInDecimal ;
        }
        getline(file, value, '.');
        getline(file, value, ';');
        if(value.length()){
            ss << value;
            ss >> hex >> numInDecimal;
            ss.clear();
            blocks[cnt].endCode = numInDecimal;
        }
        getline(file, value);
        if(value.length()){
            trim(value);
            blocks[cnt].codeName = value;
        }
        cnt++;
    }

    string input;
    char *str;
    unsigned char *pt;
    int lenptr;
    int codept;
    int readLength;
    int codeId;

    while (getline(cin, input)) {
        str = (char *)input.data();
        pt = (unsigned char *)str;
```

```
        readLength = 0;
        while(readLength < input.length()){
            lenptr = 0;
            codept = utf8_to_codepoint(pt, &lenptr);
            codeId = findId(codept);
            if(codeId != -1) {
                addOccurence(codeId);
            }
            pt += lenptr;
            readLength += lenptr;
        }
    }

    cout << blocks[findMaxId()].codeName << endl;

    return 0;
}

int findId(int codepoint){
    for (int i = 0; i < cnt; i++) {
        if (blocks[i].startCode <= codepoint &&
blocks[i].endCode >= codepoint) {
            return i;
        }
    }
    return -1;
}

void addOccurence(int key){
    if(occurenceMap.count(key)!=0){
        occurenceMap[key]++;
        maxOccurence = occurenceMap[key]>maxOccurence ?
occurenceMap[key] : maxOccurence;
    }else{
        occurenceMap[key] = 0;
    }
}

int findMaxId(){
    for(map<int, int>::iterator mapItr =
occurenceMap.begin(); mapItr != occurenceMap.end();
++mapItr){
        if (mapItr->second == maxOccurence) {
            return mapItr->first;
        }
    }
    return -1;
```

```
}

void trim(string &str){
    if(!str.empty()){
        str.erase(0,str.find_first_not_of(" "));
        str.erase(str.find_last_not_of(" ") + 1);
    }
}

unsigned int utf8_to_codepoint(const unsigned char *u, int
*lenptr) {
    // Returns 0 if something goes wrong
    // Passes back the length
    unsigned int cp = 0;

    *lenptr = 0;
    if (u) {
        if (*u < 0xc0) {
            cp = (unsigned int)*u;
            *lenptr = 1;
        } else {
            *lenptr = isutf8(u);
            if (*lenptr == 0) {
                return 0;
            }
            switch (*lenptr) {
                case 2:
                    cp = (u[0] - 192) * 64 + u[1] - 128;
                    break;
                case 3:
                    cp = (u[0] - 224) * 4096
                    + (u[1] - 128) * 64 + u[2] - 128;
                    break;
                default:
                    cp = (u[0] - 240) * 262144
                    + (u[1] - 128) * 4096
                    + (u[2] - 128) * 64 + u[3] - 128;
                    break;
            }
        }
    }
    return cp;
}

int  isutf8(const unsigned char *u) {
    // Validate utf8 character.
    // Returns the length, 0 if invalid.
```

```
    int len = 0;

    if (u) {
        if (*u < 0xc0) {
            len = 1;
        } else {
            if ((*u & 0xe0) == 0xc0) {
                // U-00000080 - U-000007FF : 110xxxxx 10xxxxxx
                len = 2;
            } else if ((*u & 0xf0) == 0xe0) {
                // U-00000800 - U-0000FFFF : 1110xxxx 10xxxxxx 10xxxxxx
                len = 3;
            } else if ((*u & 0xf8) == 0xf0) {
                // U-00010000 - U-001FFFFF : 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx
                len = 4;
            } else {
                // malformed UTF-8 character
                return 0;
            }
            // Check that the UTF-8 character is OK
            int i;
            for (i = 1; i < len; i++ ) {
                if ((u[i] & 0xC0) != 0x80) {
                    return 0;
                }
            }
        }
    }
    return len;
}
```

```
//
//  main.hpp
//  lab4
//
//  Created by 邱煜 on 2019/4/12.
//  Copyright © 2019 邱煜. All rights reserved.
//

#ifndef main_hpp
#define main_hpp

```

```
struct Blocks {
    int id;
    int startCode;
    int endCode;
    std::string codeName;
};


// return the id of the block of the corresponding codepoint
// return -1 if not found
int findId(int codepoint);

// trim the spaces in the starting and ending of the str
void trim(std::string &str);

// set the value of the key +1
// if this key not exist, add this key add set its value to 1
void addOccurence(int key);

// find the code block index which has the maximum occurence
int findMaxId(void);

// utf8_to_codepoint returns 0 if conversion fails. If it succeeds,
// the value pointed by lenptr is set to the number of bytes of the
// UTF-8 character
unsigned int utf8_to_codepoint(const unsigned char *u, int *lenptr);

// Returns the length in bytes, 0 if invalid
int isutf8(const unsigned char *u);
#endif /* main_hpp */
```

H2 Part 3 - Result & Verification

H3 Test case

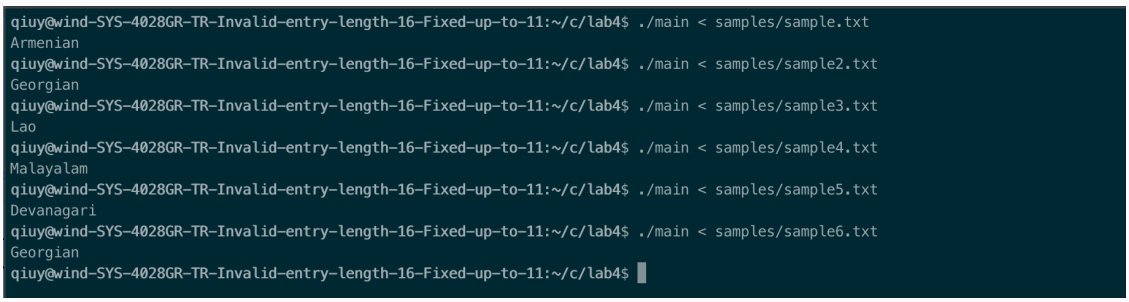
Test case #1:

```
Input:
```

```
1.  ./main < samples/sample.txt
2.  ./main < samples/sample2.txt
3.  ./main < samples/sample3.txt
4.  ./main < samples/sample4.txt
5.  ./main < samples/sample5.txt
6.  ./main < samples/sample6.txt

Output:
1.  Armenian
2.  Georgian
3.  Lao
4.  Malayalam
5.  Devanagari
6.  Georgian

Verification:
1.  Armenian
2.  Georgian
3.  Lao
4.  Malayalam
5.  Devanagari
6.  Georgian
```



H2 Part 4 - Difficulties & Solutions

1. The given files are c files. I copied the functions I used to my own files.