CME 2202 DATA ORGANIZATION AND MANAGEMENT

ASSIGNMENT-1 REPORT

SAMPLE SCREENSHOTS

First, we created a struct with various variables. These structs keep the attributes of the people from the input file. We used these structs to easily read the data from the input file using the “fread()” function of C language.

```
struct record
{
    char name[64];    //utf16
    char surname[32]; //utf8
    char gender;
    char email[32];
    char phone_number[16];
    char address[32];
    char level_of_education[8];
    unsigned int income_level; // given little-endian
    unsigned int expenditure;  // given big-endian
    char currency_unit[16];
    char currentMood[32];
    float height;
    unsigned int weight;
};
struct headers
{
    char name[64];    //utf16
    char surname[32]; //utf8
    char gender;
    char email[32];
    char phone_number[16];
    char address[32];
    char level_of_education[8];
    unsigned int income_level; // given little-endian
    unsigned int expenditure;  // given big-endian
    char currency_unit[16];
    char currentMood[32];
    float height;
    unsigned int weight;
};
```

Then we created a function for endian transformations.

```c
unsigned int tobigEndian(unsigned int number)
{
    number = ((number >> 24) & 0xff) |      // move byte 3 to byte 0
             ((number << 8) & 0xff0000) |   // move byte 1 to byte 2
             ((number >> 8) & 0xff00) |     // move byte 2 to byte 1
             ((number << 24) & 0xff000000); // byte 0 to byte 3
    return number;
}
```

After creating the necessary structs and functions we started to code our algorithm. Firstly, we opened the file with UTF-16 encoding and counted all the person data and printed the names of all the people on the input file to the screen.

```c
if ((finp = fopen(argv[1], "rb,ccs=UTF-16LE")) == NULL) //open file with UTF-16
{
    printf("\nError! File cannot be opened.\n"); //display error message if file doesnt exists
    exit(1);
}
fout = fopen(argv[2], "w");
fread(&header, sizeof(struct headers), 1, finp); //read the headers
printf("\nName list with UTF-16:\n");
do
{
    fread(&line, sizeof(struct record), 1, finp); //read a line to a struct
    printf("%s\n", line.name);
    linecount = linecount + 1;
} while (line.height != 0);
fclose(finp);
```

After we had the count of the amount of people on the input file, we used this number to determine the repetition count of our main loop which prints the data into the .xml file. We used the header struct to print the headers of the attributes and used the line struct to print the actual data.

```c
for (int i = 1; i < linecount; i++)
{
    fread(&line, sizeof(struct record), 1, finp); //read a line to a struct
    fprintf(fout, "\t<row id=\"%d\">\n", i);
    fprintf(fout, "\t\t<%s>%s</%s>\n", header.name, line.name, header.name);
    fprintf(fout, "\t\t<%s>%s</%s>\n", header.surname, line.surname, header.surname);
    //correct g as gender
    if (header.gender = 'g')
    {
        fprintf(fout, "\t\t<gender>%c</gender>\n", line.gender);
    }
    else
    {
        fprintf(fout, "\t\t<%s>%c</%s>\n", header.gender, line.gender, header.gender);
    }
    fprintf(fout, "\t\t<%s>%s</%s>\n", header.email, line.email, header.email);
    fprintf(fout, "\t\t<%s>%s</%s>\n", header.phone_number, line.phone_number, header.phone_number);
    fprintf(fout, "\t\t<%s>%s</%s>\n", header.address, line.address, header.address);
    //correct level_of_ed as level_of_education
    if (strcmp(header.level_of_education, "level_of_ed") == 0)
    {
        fprintf(fout, "\t\t<level_of_education>%s</level_of_education>\n", line.level_of_education);
    }
    else
    {
        fprintf(fout, "\t\t<%s>%s</%s>\n", header.level_of_education, line.level_of_education, header.level_of_education);
    }
    fprintf(fout, "\t\t<income_level bigEnd=\"%u\">%d</income_level>\n", tobigEndian(line.income_level), line.income_level);
    line.expenditure = tobigEndian(line.expenditure);
    fprintf(fout, "\t\t<expenditure bigEnd=\"%u\">%d</expenditure>\n", tobigEndian(line.expenditure), line.expenditure);
    fprintf(fout, "\t\t<%s>%s</%s>\n", header.currency_unit, line.currency_unit, header.currency_unit);
    fprintf(fout, "\t\t<%s>%s</%s>\n", header.currentMood, line.currentMood, header.currentMood);
    fprintf(fout, "\t\t<height>%.2f</height>\n", line.height);
    fprintf(fout, "\t\t<weight>%d</weight>\n", line.weight);
    fprintf(fout, "</row>\n\n");
}
```

PROBLEMS

We could not read records.dat. It could not find the file all the time. We realized that the source of this problem was due to misspelling. We solved this problem with a small fix.

We couldn’t code the .xsd validiton part because we couldn’t not enough time.