ECEN449-HW2

Suqian Wang
825009505

1.

```c
//
//  main.c
//  problem1
//
//  Created by Susan Wang on 10/8/17.
//  Copyright © 2017 Suqian Wang. All rights reserved.
//

#include <stdio.h>
#include <stdlib.h>

int cmpfunc(const void* a, const void* b) {
    return ( *(int*)b - *(int*)a );
}

int main() {
    // file stream for input file
    FILE *in_file = NULL;
    in_file = fopen("input1.txt", "r");
    if (in_file == NULL) {
        printf("Error opening file!\n");
        exit(1);
    }


    // read line by line from file and stored in an integer array
    int i = 0;
    char line[5];
    int number_array[10];
    while (fgets(line, sizeof(line), in_file) != NULL) {
        number_array[i] = atoi(line);
        i++;
    }
    fclose(in_file);

    // sort numbers in array in desending order
    int n_item = i;
    qsort(number_array, n_item, sizeof(int), cmpfunc);

    // file stream for output file
    FILE *out_file = NULL;
    out_file = fopen("output1.txt", "w");
    if (out_file == NULL) {
        printf("Error opening file!\n");
        exit(1);
    }

    // wrtie to each item in the array to the output file
    for (int i = 0; i < n_item; i++) {
        fprintf(out_file, "%d\n", number_array[i]);
    }
    fclose(out_file);

    return 0;
}
```

2.

```
//
//  main.c
//  problem2
//
//  Created by Susan Wang on 10/8/17.
//  Copyright © 2017 Suqian Wang. All rights reserved.
//

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// a structure of type "Record"
```

```
typedef struct {
    int id;
    int credit;
    float gpa;
} Record;

// compare function for qsort argument
int cmpfunc_credit(const void* a, const void* b) {
    int l = ((Record *)a)->credit;
    int r = ((Record *)b)->credit;
    return (r - l);
}

int cmpfunc_gpa(const void* a, const void* b) {
    float l = ((Record *)a)->gpa;
    float r = ((Record *)b)->gpa;
    float diff = r - l;
    if (diff < 0) {
        return -1;
    }
    else if (diff > 0) {
        return 1;
    }
    else {
        return 0;
    }
}

// split the line string and save to corresponding category
Record split_string(char* line_ptr) {
    Record record;
    const char s[2] = " ";
    record.id = atoi(strtok(line_ptr, s));
    record.credit = atoi(strtok(NULL, s));
    record.gpa = atof(strtok(NULL, s));
    return record;
}

int main() {
    // file stream for input file
    FILE *in_file = NULL;
    in_file = fopen("input2.txt", "r");
    if (in_file == NULL) {
        printf("Error opening file!\n");
        exit(1);
    }

    // read line by line from file and stored records in an array
    char line[30];
    char *first_line = fgets(line, sizeof(line), in_file);
    int num_record = atoi(first_line);
    Record record_array[num_record];

    for (int i = 0; i < num_record; i++) {
        record_array[i] = split_string(fgets(line, sizeof(line), in_file));
    }

    // sort records in desending order by credit or gpa
    char *flag = fgets(line, sizeof(line), in_file);

    fclose(in_file);

    if (*flag == 'C') {
        qsort(record_array, num_record, sizeof(Record), cmpfunc_credit);
    }
    if (*flag == 'G') {
        qsort(record_array, num_record, sizeof(Record), cmpfunc_gpa);
    }

    // file stream for output file
    FILE *out_file = NULL;
    out_file = fopen("output2.txt", "w");
    if (out_file == NULL) {
        printf("Error opening file!\n");
        exit(1);
    }

    // wrtie sorted record to the output file
    for (int i = 0; i < num_record; i++) {
        fprintf(out_file, "%d %d %f\n", record_array[i].id, record_array[i].credit, record_array[i].gpa);
    }
    fclose(out_file);

    return 0;
}
```

3.

```
//
//  main.c
//  problem3
//
//  Created by Susan Wang on 10/9/17.
//  Copyright © 2017 Suqian Wang. All rights reserved.
//
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void swap(void *p1, void *p2, unsigned int size) {
    // allocate temporary memory
    void *temp = malloc(size);
    // copy size characters from memory area p1 to temporary memory area
    memcpy(temp, p1, size);
    memcpy(p1, p2, size);
    memcpy(p2, temp, size);
    // free allocated temporary memory after done with it
    free(temp);
}

int main() {


    int x = 10; int y = 20;
    swap(&x, &y, sizeof(int));
    //x = 20, y = 10 after swap
    printf("x = %d, y = %d\n", x, y);


    char a = 'F'; char b = 'G';
    swap(&a, &b, sizeof(char));
    //a = G, b = F after swap
    printf("a = %c, b = %c", a, b);


    return 0;
}
```

4.

```
# Makefile

CXX = g++
CXXFLAGS = -Wall -g

all: main

main: main.o transport.o TCP.o UDP.o
	$(CXX) $(CXXFLAGS) -o main main.o transport.o TCP.o UDP.o

main.o: main.cpp transport.hpp TCP.hpp UDP.hpp
	$(CXX) $(CXXFLAGS) -c main.cpp

transport.o: transport.cpp transport.hpp TCP.hpp UDP.hpp
	$(CXX) $(CXXFLAGS) -c transport.cpp

TCP.o: TCP.cpp TCP.hpp
	$(CXX) $(CXXFLAGS) -c TCP.cpp

UDP.o: UDP.cpp UDP.hpp
	$(CXX) $(CXXFLAGS) -c UDP.cpp

clean:
	rm -f *.o main
```