```
//Edward Yang
//
//  QuickSort_Skeleton.cpp
//
//  Created by Bahamon, Julio on 6/25/19.
//  UNC Charlotte
//  Copyright © 2019 Bahamon, Julio. All rights reserved.
//

#include <iostream>
#include <cstdlib>
#include <cstring>

using namespace std;

//  Declaring a new struct to store patient data
struct patient
{
    int age;
    char name[20];
    float balance;
};

//  TODO:
//  IMPLEMENT A FUNCTION THAT COMPARES TWO PATIENTS BY AGE

//  THE FUNCTION RETURNS AN INTEGER AS FOLLOWS:
//      -1 IF THE AGE OF THE FIRST PATIENT IS LESS
//         THAN THE SECOND PATIENT'S AGE
//       0 IF THE AGES ARE EQUAL
```

```
//       1 OTHERWISE

int compareByAge(const void *a, const void *b)
{
    patient *patientA = (patient *)a;
    patient *patientB = (patient *)b;
    if (patientA->age < patientB->age)
        return -1;
    if (patientA->age == patientB->age)
        return 0;
    return 1;
}

//  TODO:
//  IMPLEMENT A FUNCTION THAT COMPARES TWO PATIENTS BY BALANCE DUE

//  THE FUNCTION RETURNS AN INTEGER AS FOLLOWS:
//      -1 IF THE BALANCE FOR THE FIRST PATIENT IS LESS
//         THAN THE SECOND PATIENT'S BALANCE
//       0 IF THE BALANCES ARE EQUAL
//       1 OTHERWISE

int compareByBalance(const void *a, const void *b)
{
    patient *patientA = (patient *)a;
    patient *patientB = (patient *)b;
    if (patientA->balance < patientB->balance)
        return -1;
    if (patientA->balance == patientB->balance)
        return 0;
```

```
    return 1;
}

//  TODO:
//  IMPLEMENT A FUNCTION THAT COMPARES TWO PATIENTS BY NAME

//  THE FUNCTION RETURNS AN INTEGER AS FOLLOWS:
//      -1 IF THE NAME OF THE FIRST PATIENT GOES BEFORE
//         THE SECOND PATIENT'S NAME
//       0 IF THE AGES ARE EQUAL
//       1 OTHERWISE
//
//  HINT: USE THE strncmp FUNCTION
//  (SEE http://www.cplusplus.com/reference/cstring/strncmp/)

int compareByName(const void *a, const void *b)
{
    patient *patientA = (patient *)a;
    patient *patientB = (patient *)b;
    return strncmp(patientA->name, patientB->name, 20);
}

// display the contents of the array
void displayPatients(patient *patientList, int total)
{
    for (int i = 0; i < total; ++i)
    {
        cout << "Name: " << patientList[i].name
             << ", Age: " << patientList[i].age
             << ", Balance: " << patientList[i].balance
```

```
             << endl;
    }
}

//  The main program
int main()
{
    int total_patients = 6;

    // Storing some test data
    struct patient patient_list[6] = {
        {25, "Juan_Valdez", 1250},
        {15, "James_Morris", 2100},
        {32, "Tyra_Banks", 750},
        {62, "Mario_O'Donell", 375},
        {53, "Pablo_Picasso", 615},
        {0, "", 0.0}
    };

    // New entry
    cout << "Please enter your last name (use underscores for spaces): ";
    cin >> patient_list[5].name;

    cout << "Enter age: ";
    cin >> patient_list[5].age;

    cout << "Enter balance due: ";
    cin >> patient_list[5].balance;

    //  TODO:
```

```
//  IMPLEMENT THE CODE TO DISPLAY THE CONTENTS
//  OF THE ARRAY BEFORE SORTING

cout << "Patient List: " << endl;
displayPatients(patient_list, total_patients);
cout << endl;

cout << "Sorting..." << endl;

//  TODO:
//  CALL THE qsort FUNCTION TO SORT THE ARRAY BY PATIENT AGE
qsort(patient_list, total_patients, sizeof(patient), compareByAge);

//  TODO:
//  DISPLAY THE CONTENTS OF THE ARRAY
//  AFTER SORTING BY AGE
cout << "Patient List - Sorted by Age: " << endl;
displayPatients(patient_list, total_patients);
cout << endl;

cout << "Sorting..." << endl;



//  TODO:
//  CALL THE qsort FUNCTION TO SORT THE ARRAY BY PATIENT BALANCE

qsort(patient_list, total_patients, sizeof(patient), compareByBalance);

//  TODO:
//  DISPLAY THE CONTENTS OF THE ARRAY
```

```
    //  AFTER SORTING BY BALANCE
    cout << "Patient List - Sorted by Balance Due: " << endl;
    displayPatients(patient_list, total_patients);
    cout << endl;

    cout << "Sorting..." << endl;


    //  TODO:
    //  CALL THE qsort FUNCTION TO SORT THE ARRAY BY PATIENT NAME

    qsort(patient_list, total_patients, sizeof(patient), compareByName);

    //  TODO:
    //  DISPLAY THE CONTENTS OF THE ARRAY
    //  AFTER SORTING BY NAME

    cout << "Patient List - Sorted by Name: " << endl;
    displayPatients(patient_list, total_patients);

    cout << endl;

    return 0;
}



//Edward Yang
#include <iostream>
```

```
int main() {
    // Declare and assign values to variables
    int myInt = 15;
    int* myPointer = &myInt;

    // Print the memory address of myInt and the value contained in myPointer
    std::cout << "Memory address of myInt: " << &myInt << std::endl;
    std::cout << "Value contained in myPointer: " << *myPointer << std::endl;

    // Print the value of myInt and the value pointed to by myPointer
    std::cout << "Value of myInt: " << myInt << std::endl;
    std::cout << "Value pointed to by myPointer: " << *myPointer << std::endl;

    // Change values and print to console
    myInt = 10;

    // Repeat the printing steps
    std::cout << "Memory address myInt after: " << &myInt << std::endl;
    std::cout << "Value in myPointer after: " << *myPointer << std::endl;
    std::cout << "Value of myInt after: " << myInt << std::endl;
    std::cout << "Value pointed to by myPointer after: " << *myPointer <<
std::endl;

    return 0;
}



//
//  Processes.cpp
//  ITSC 3146
//
```

```
//  Created by Bahamon, Julio on 1/12/17.
//

/*
 @file Processes.cpp
 @author student name, student@uncc.edu
 @author student name, student@uncc.edu
 @author student name, student@uncc.edu
 @description: <ADD DESCRIPTION>
 @course: ITSC 3146
 @assignment: in-class activity [n]
 */

#ifndef Processes_cpp
#define Processes_cpp

#include "Processes.h"

using namespace std;

// Part 1: Working With Process IDs
pid_t getProcessID(void)
{
   // Returns the current process ID
   return getpid();
}

// Part 2: Working With Multiple Processes
string createNewProcess(void)
{
```

```
    pid_t id = fork();

    // DO NOT CHANGE THIS LINE OF CODE
    process_id = id;

    if (id == -1)
    {
        return "Error creating process";
    }
    else if (id == 0)
    {
        // TODO: Add your code here (child)
        cout << "I am a child process!\n";
        return "I am bored of my parent, switching programs now";
    }
    else
    {
        // TODO: Add your code here (parent)
        cout << "I just became a parent!\n";

        int status;
        wait(&status); // Wait for child process to terminate

        return "My child process just terminated!";
    }
}

// Part 3: Working With External Commands"
void replaceProcess(char * args[])
{
```

```
    // Spawn a process to execute the user's command.
    pid_t id = fork();

    // TODO: Add your code here

    if (id == -1)
    {
        // Handle error in fork
        perror("fork");
        exit(EXIT_FAILURE);
    }
    else if (id == 0)
    {
        // Child process
        if (execvp(args[0], args) == -1)
        {
            // Handle error in execvp
            perror("execvp");
            exit(EXIT_FAILURE);
        }
    }
    else
    {
        // Parent process
        int status;
        wait(&status); // Wait for child process to terminate
        exit(EXIT_SUCCESS);
    }
}
```

```
#endif /* TestProg_cpp */



//edward yang
#include <iostream>
using namespace std;

int main() {
    const int SIZE = 4;
    int my_ints[SIZE] = {25,7,1,10}; // Pre-defined values
    int* my_ptrs[SIZE];

    // Initialize my_ptrs to point to my_ints elements
    for (int i = 0; i < SIZE; ++i) {
        my_ptrs[i] = &my_ints[i];
    }

    // Bubble Sort: Sort the pointers in my_ptrs based on the values they point
to
    for (int i = 0; i < SIZE - 1; ++i) {
        for (int j = 0; j < SIZE - i - 1; ++j) {
            if (*my_ptrs[j] > *my_ptrs[j + 1]) {
                // Swap pointers
                int* temp = my_ptrs[j];
                my_ptrs[j] = my_ptrs[j + 1];
                my_ptrs[j + 1] = temp;
            }
        }
    }

    // Print the values pointed to by my_ptrs
```

```
    cout << "Sorted values:" << endl;
    for (int i = 0; i < SIZE; ++i) {
        cout << *my_ptrs[i] << " ";
    }
    cout << endl;

    // Sanity check: Print my_ints to ensure it's unchanged
    cout << "Original array (my_ints):" << endl;
    for (int i = 0; i < SIZE; ++i) {
        cout << my_ints[i] << " ";
    }
    cout << endl;

    return 0;
}
```