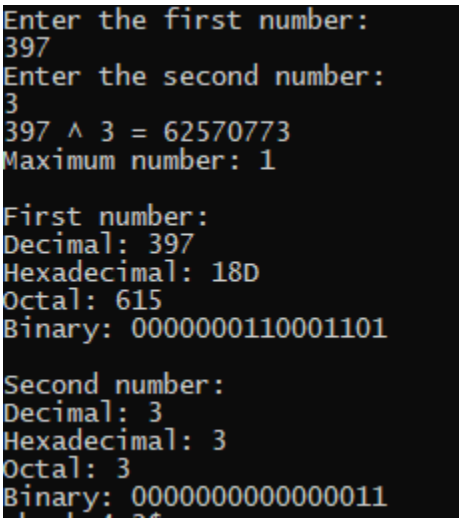
Saul Reyna
EECE2160

Homework 1

Assignment 1




```cpp
#include <iostream>
#include <string>
#include <math.h>
#include <algorithm>
#include <bitset>

using namespace std;

int to_power_of(int n1, int n2) {
    return pow(n1, n2);
}
bool check_max(int n1, int n2) {
    if (n1 > n2) {
        return n1;
    } else return n2;
}

void dec2Hex(int n) {
    string conv = "";
    int temp;
    char hex[]={'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};
```

```
while(n > 0) {
        temp = n % 16;
        conv = hex[temp] + conv;
        n = n / 16;
    }

    cout << "Hexadecimal: " << conv << endl;
}

void dec2Oct(int n) {
    string conv = "";
    int temp;

    while(n > 0) {
        temp = n % 8;
        conv = to_string(temp) + conv;
        n = n / 8;
    }

    cout << "Octal: " << conv << endl;
}

void dec2Bin(int n) {
    bitset<16> bits;
    int temp;

    for (int i = 0; i < 16; i++) {
        temp = n % 2;
        if(temp == 1) {
            bits.set(i);
        }
        n = n / 2;
    }

    cout << "Binary: " << bits.to_string() << endl;
}

void convert(int n1) {
    cout << "Decimal: " << n1 << endl;
    dec2Hex(n1);
    dec2Oct(n1);
    dec2Bin(n1);
}

int main() {
```

```
    int number1, number2;

    cout << "Enter the first number: " << endl;
    cin >> number1;
    cout << "Enter the second number: " << endl;
    cin >> number2;
    cout << number1 << " ^ " << number2 << " = " << to_power_of(number1, number2)
 << endl;
    cout << "Maximum number: " << check_max(number1, number2) << endl << endl;

    cout << "First number: " << endl;
    convert(number1);
    cout << endl;
    cout << "Second number: " << endl;
    convert(number2);
}
```

Assignment 2

```
#include <iostream>
#include <fstream>
#include <string>
#include <cstdlib>

using namespace std;

struct Car {
    string make;
    string model;
    int year;
    string color;
};

Car * insert_array(Car *c, string file_path) {
    typedef struct Car Car;
    string line;
    ifstream file;
    file.open(file_path);
    int i = 0;

    while(!file.eof()){
        getline(file, line);
        string make = line.substr(0, line.find(",", 0));
        string rest = line.substr(line.find(",") + 2);
```

```
        string model = rest.substr(0, rest.find(","));
        rest = rest.substr(rest.find(",") + 2);
        int year = stoi(rest.substr(0, rest.find(",")));
        rest = rest.substr(rest.find(",")+2);
        string color = rest;
        Car curr_car = {make, model, year, color};
        c[i] = curr_car;
        i++;
    }

    file.close();
    return c;
}

void print_car(Car c) {
    cout << "\t Make: " << c.make << endl;
    cout << "\t Model: " << c.model << endl;
    cout << "\t Year: " << c.year << endl;
    cout << "\t Color: " << c.color << endl;
}

void print_cars_array(Car* c, int length) {
    for (int i = 0; i < length; i++) {
        cout << "Car " << (i + 1) << ": " << endl;
        print_car(c[i]);
    }
}

Car * sort_cars_by_make(Car* c, int length) {
    Car* temp = new Car[length]();

    int i, min_i;
    string min_str;

    for (i = 0; i < length-1; i++) {
        min_i = i;
        min_str = c[i].make;
        for (int j = i+1; j < length; j++) {
            if (c[j].make < min_str) {
                min_str = c[j].make;
                min_i = j;
            }
        }
        c[min_i] = c[i];
        c[i].make = min_str;
```

```
    }

    return c;
}

bool operator ==(Car car1, Car car2) {
    return (car1.make == car2.make) && (car1.model == car2.model)
        && (car1.year == car2.year) && (car1.color == car2.color);
}

void print_duplicates(Car* c, int length) {
    cout << "Duplicate Cars: " << endl;
    for (int i = 0; i < length; i++) {
        for (int j = 0; j < length; j++) {
            if (j != i) {
                if(c[i] == c[j]) {
                    print_car(c[i]);
                    cout << endl;
                }
            }
        }
    }
}

void save_cars_in_file(Car* c, int length) {
    ofstream file ("CarRecords.txt");

    if(file.is_open()) {
        for(int i = 0; i < length; i++) {
            string line = "";

            if (i == (length - 1)) {
                line = c[i].make + ", " + c[i].model + ", " +
                to_string(c[i].year) + ", " + c[i].color;
            }else {
                line = c[i].make + ", " + c[i].model + ", " +
                to_string(c[i].year) + ", " + c[i].color + "\n";
            }
            file << line;
        }
    }

    file.close();
    cout << "Done writing to file" << endl;
}
```

Saul Reyna
EECE2160

```
int main() {
    int total_lines = 0;
    string line;
    ifstream file;
    file.open("CarRecords.txt");

    while(!file.eof()) {
        getline(file, line);
        total_lines++;
    }

    Car* cars = new Car[total_lines]();
    int input;

    while (true) {
        cout << "MENU - Select an option: " << endl;
        cout << "1. Insert car records into an array" << endl;
        cout << "2. Print the cars array" << endl;
        cout << "3. Sort cars by make" << endl;
        cout << "4. Print duplicates" << endl;
        cout << "5. Save car records into a file" << endl;
        cout << "6. Exit" << endl;
        cin >> input;
        switch(input) {
            case 1:
                cars = insert_array(cars, "CarRecords.txt");
                cout << "Car Records loaded from CarRecords.txt" << endl;
                cout << "Total lines: " << total_lines << endl;
                break;
            case 2:
                print_cars_array(cars, total_lines);
                break;
            case 3:
                sort_cars_by_make(cars, total_lines);
                cout << "Cars sorted by make" << endl;
                break;
            case 4:
                print_duplicates(cars, total_lines);
                break;
            case 5:
                save_cars_in_file(cars, total_lines);
                break;
            case 6:
                return 0;
```

```
                break;
            default:
                cout << "Not an available option" << endl;
        }
    }
}
```

```
MENU - Select an option:
1. Insert car records into an array
2. Print the cars array
3. Sort cars by make
4. Print duplicates
5. Save car records into a file
6. Exit
1
Car Records loaded from CarRecords.txt
Total lines: 10
```

```
2
Car 1:
        Make: Subaru
        Model: Outback
        Year: 2016
        Color: green
Car 2:
        Make: Toyota
        Model: Corolla
        Year: 2006
        Color: white
Car 3:
        Make: Dodge
        Model: Neon
        Year: 1993
        Color: pink
Car 4:
        Make: Ford
        Model: Fusion
        Year: 2013
        Color: yellow
Car 5:
        Make: Honda
        Model: Fit
        Year: 2015
        Color: blue
Car 6:
        Make: Ford
        Model: Expedition
        Year: 2009
        Color: silver
Car 7:
        Make: Toyota
        Model: Corolla
        Year: 2006
        Color: white
Car 8:
        Make: Ford
        Model: Fusion
        Year: 2013
        Color: yellow
Car 9:
        Make: Jeep
        Model: Cherokee
        Year: 1999
        Color: red
Car 10:
        Make: Mazda
        Model: Protoge
        Year: 1996
        Color: gold
```

```
MENU - Select an option:
1. Insert car records into an array
2. Print the cars array
3. Sort cars by make
4. Print duplicates
5. Save car records into a file
6. Exit
3
Cars sorted by make
MENU - Select an option:
1. Insert car records into an array
2. Print the cars array
3. Sort cars by make
4. Print duplicates
5. Save car records into a file
6. Exit
2
Car 1:
        Make: Dodge
        Model: Outback
        Year: 2016
        Color: green
Car 2:
        Make: Ford
        Model: Corolla
        Year: 2006
        Color: white
Car 3:
        Make: Ford
        Model: Outback
        Year: 2016
        Color: green
Car 4:
        Make: Ford
        Model: Corolla
        Year: 2006
        Color: white
Car 5:
        Make: Honda
        Model: Fit
        Year: 2015
        Color: blue
Car 6:
        Make: Jeep
        Model: Outback
        Year: 2016
        Color: green
Car 7:
        Make: Mazda
        Model: Corolla
        Year: 2006
```

```
Car 8:
        Make: Subaru
        Model: Corolla
        Year: 2006
        Color: white
Car 9:
        Make: Toyota
        Model: Corolla
        Year: 2006
        Color: white
Car 10:
        Make: Toyota
        Model: Corolla
        Year: 2006
        Color: white
```

```
4
Duplicate Cars:
        Make: Ford
        Model: Corolla
        Year: 2006
        Color: white

        Make: Ford
        Model: Corolla
        Year: 2006
        Color: white

        Make: Toyota
        Model: Corolla
        Year: 2006
        Color: white

        Make: Toyota
        Model: Corolla
        Year: 2006
        Color: white
```

```
MENU - Select an option:
1. Insert car records into an array
2. Print the cars array
3. Sort cars by make
4. Print duplicates
5. Save car records into a file
6. Exit
5
Done writing to file
MENU - Select an option:
1. Insert car records into an array
2. Print the cars array
3. Sort cars by make
4. Print duplicates
5. Save car records into a file
6. Exit
6
```

Saul Reyna
EECE2160

```
1. Insert car records into an array
2. Print the cars array
3. Sort cars by make
4. Print duplicates
5. Save car records into a file
6. Exit
1
Car Records loaded from CarRecords.txt
Total lines: 10
MENU - Select an option:
1. Insert car records into an array
2. Print the cars array
3. Sort cars by make
4. Print duplicates
5. Save car records into a file
6. Exit
2
Car 1:
        Make: Dodge
        Model: Outback
        Year: 2016
        Color: green
Car 2:
        Make: Ford
        Model: Corolla
        Year: 2006
        Color: white
Car 3:
        Make: Ford
        Model: Outback
        Year: 2016
        Color: green
Car 4:
        Make: Ford
        Model: Corolla
        Year: 2006
        Color: white
Car 5:
        Make: Honda
        Model: Fit
        Year: 2015
        Color: blue
Car 6:
        Make: Jeep
        Model: Outback
        Year: 2016
        Color: green
Car 7:
        Make: Mazda
        Model: Corolla
        Year: 2006
```

```
Car 8:
        Make: Subaru
        Model: Corolla
        Year: 2006
        Color: white
Car 9:
        Make: Toyota
        Model: Corolla
        Year: 2006
        Color: white
Car 10:
        Make: Toyota
        Model: Corolla
        Year: 2006
        Color: white
```

Assignment 3

```
MENU - Select an option:
1. Insert Linked List
2. Print Cars List
3. Exit
1
Records have been inserted into a linked list
```

Saul Reyna
EECE2160

```
2
        Make: Toyota
        Model: Corolla
        Year: 2006
        Color: white

        Make: Toyota
        Model: Corolla
        Year: 2006
        Color: white

        Make: Subaru
        Model: Corolla
        Year: 2006
        Color: white

        Make: Mazda
        Model: Corolla
        Year: 2006
        Color: white

        Make: Jeep
        Model: Outback
        Year: 2016
        Color: green

        Make: Honda
        Model: Fit
        Year: 2015
        Color: blue
```

```
        Make: Ford
        Model: Corolla
        Year: 2006
        Color: white

        Make: Ford
        Model: Outback
        Year: 2016
        Color: green

        Make: Ford
        Model: Corolla
        Year: 2006
        Color: white

        Make: Dodge
        Model: Outback
        Year: 2016
        Color: green
```

```cpp
#include <iostream>
#include <fstream>
#include <string>

using namespace std;

struct Car {
    string make;
    string model;
    int year;
    string color;
};

struct car_node {
    Car current_car;
```

```
    car_node *next_car;
};

car_node * insert_linkedList(car_node* cn, string file_path) {
    string line;
    ifstream file;
    file.open(file_path);

    while(!file.eof()) {
        car_node *temp = new car_node;
        getline(file, line);

        string make = line.substr(0, line.find(",", 0));
        string rest = line.substr(line.find(",") + 2);
        string model = rest.substr(0, rest.find(","));
        rest = rest.substr(rest.find(",") + 2);
        int year = stoi(rest.substr(0, rest.find(",")));
        rest = rest.substr(rest.find(",")+2);
        string color = rest;
        Car curr_car = {make, model, year, color};

        temp->current_car = curr_car;
        temp->next_car = cn;
        cn = temp;
    }

    file.close();
    return cn;

}

void print_cars_list(car_node* cn) {
    if (cn->next_car == NULL) {

    }
    else {
        cout << "\tMake: " << cn->current_car.make << endl;
        cout << "\tModel: " << cn->current_car.model << endl;
        cout << "\tYear: " << cn->current_car.year << endl;
        cout << "\tColor: " << cn->current_car.color << endl << endl;;
        print_cars_list(cn->next_car);
    }
}

int main() {
```

```
    typedef struct Car Car;
    car_node *ptr = new car_node;
    ptr->next_car = NULL;
    int input;

    while(true) {
        cout << "MENU - Select an option: " << endl;
        cout << "1. Insert Linked List" << endl;
        cout << "2. Print Cars List" << endl;
        cout << "3. Exit" << endl;
        cin >> input;
        switch(input) {
            case 1:
                ptr = insert_linkedList(ptr, "CarRecords.txt");
                cout << "Records have been inserted into a linked list" << endl;
                break;
            case 2:
                print_cars_list(ptr);
                break;
            case 3:
                return 0;
                break;
        }
    }

    delete ptr;
}
```