**Homework 36**

**Topic: Branching Statements and Logical Operators (exercises 1 to 6)**

//------------------------------------------------------------------

// File name: Exercise\_1.cpp

// Assign ID:

// Due Date: 18/06/24 at 11pm

//

// Purpose: Determines the lucky number.

//

// Author: Mr. KEO Sopahnit

//------------------------------------------------------------------

#include <iostream>

using namespace std;

int main() {

//1. Store

int sixDigit, digit1, digit2, digit3, digit4, digit5, digit6;

int firstThreeSum, lastThreeSum;

string luckyNumber;

const int TEN = 10;

const int LOW\_ERROR = 100000;

const int HIGH\_ERROR = 999999;

bool error = false;

//2. Input

cout << "Enter a six-digit integer: ";

cin >> sixDigit;

//3. Process

// Error

if (sixDigit < LOW\_ERROR || sixDigit > HIGH\_ERROR) {

error = true;

}

// Extraction Digit

digit6 = sixDigit % TEN;

sixDigit /= TEN;

digit5 = sixDigit % TEN;

sixDigit /= TEN;

digit4 = sixDigit % TEN;

sixDigit /= TEN;

digit3 = sixDigit % TEN;

sixDigit /= TEN;

digit2 = sixDigit % TEN;

sixDigit /= TEN;

digit1 = sixDigit;

// Define The Lucky Number

firstThreeSum = digit1+digit2+digit3;

lastThreeSum = digit4+digit5+digit6;

luckyNumber = (firstThreeSum == lastThreeSum) ? "LUCKY" : "not LUCKY";

//4. Output

if(error){

cout << "Invalid input. Please enter a six-digit integer." << endl;

}else{

cout << " The Number is "<<luckyNumber<<endl;

}

return 0;

}

//------------------------------------------------------------------

// File name: Exercise\_2.cpp

// Assign ID:

// Due Date: 18/06/24 at 11pm

//

// Purpose: Change Order of four digit.

//

// Author: Mr. KEO Sopahnit

//------------------------------------------------------------------

#include <iostream>

using namespace std;

int main(){

//1. Store

int number, digit1, digit2, digit3, digit4;

const int TEN = 10;

const int LOW\_ERROR = 1000;

const int HIGH\_ERROR = 9999;

bool error = false;

//2.Input

cout<<"Enter four Digit: ";

cin>>number;

//3. Process

if (number < LOW\_ERROR || number > HIGH\_ERROR) {

error = true;

}

//3.1 Extact number

digit4 = number%TEN;

number /= TEN;

digit3 = number%TEN;

number /= TEN;

digit2 = number%TEN;

number /= TEN;

digit1 = number%TEN;

//4. Output

if (error){

cout<<"Invalid input. Please enter a six-digit integer."<<endl;

}

else{

cout<<"The number is: "<<digit2<<digit1<<digit4<<digit3<<endl;

}

return 0;

}

//------------------------------------------------------------------

// File name: Exercise\_3.cpp

// Assign ID:

// Due Date: 18/06/24 at 11pm

//

// Purpose: Ditermine the Min and Max.

//

// Author: Mr. KEO Sopahnit

//------------------------------------------------------------------

#include <iostream>

using namespace std;

int main(){

//1. Store

double number1, number2, number3, number4, number5, number6, number7;

double min,max;

//2. Input

cout<<"Enter seven Number: ";

cin>>number1>>number2>>number3>>number4>>number5>>number6>>number7 ;

//3. Process

max = (number1 > number2) ? number1 : number2;

max = (max > number3) ? max : number3;

max = (max > number4) ? max : number4;

max = (max > number5) ? max : number5;

max = (max > number6) ? max : number6;

max = (max > number7) ? max : number7;

min = (number1 < number2) ? number1 : number2;

min = (min < number3) ? min : number3;

min = (min < number4) ? min : number4;

min = (min < number5) ? min : number5;

min = (min < number6) ? min : number6;

min = (min < number7) ? min : number7;

//4. Output

cout<<"The max is: "<<max<<endl;

cout<<"The min is: "<<min<<endl;

return 0;

}

//------------------------------------------------------------------

// File name: Exercise\_4.cpp

// Assign ID:

// Due Date: 18/06/24 at 11pm

//

// Purpose: Minimum amount of fuel is necessary for refueling the aircraft.

//

// Author: Mr. KEO Sopahnit

//------------------------------------------------------------------

#include <iostream>

using namespace std;

int main() {

//1. Store

double distanceAB, distanceBC;

double cargoWeight;

double refuelingAmount;

double fuelConsumption, fuelConsumptionAB, fuelConsumptionBC;

const int MAX\_FUEL = 300;

enum WEIGHT { WEIGHT500 = 500, WEIGHT1000 = 1000, WEIGHT1500 = 1500, WEIGHT2000 = 2000 };

enum CONSUMPTION{ CONSUM1 = 1, CONSUM4 = 4, CONSUM7 = 7, CONSUM9 = 9};

int flag ;

//2. Input and validation

cout << "Enter the distance from point A to B (in km): ";

cin >> distanceAB;

cout << "Enter the distance from point B to C (in km): ";

cin >> distanceBC;

cout << "Enter the weight of cargo (in kg): ";

cin >> cargoWeight;

//3. Process

if (distanceAB <= 0 || distanceBC <= 0) {

cout << "Invalid distance. Distance must be greater than zero." << endl;

exit(1);

}

// Determine fuel consumption per km

if (cargoWeight <= WEIGHT500) {

fuelConsumption = CONSUM1;

} else if (cargoWeight <= WEIGHT1000) {

fuelConsumption = CONSUM4;

} else if (cargoWeight <= WEIGHT1500) {

fuelConsumption = CONSUM7;

} else if (cargoWeight <= WEIGHT2000) {

fuelConsumption = CONSUM9;

} else {

flag == 0;

}

// Calculate fuel consumption

fuelConsumptionAB = fuelConsumption \* distanceAB;

fuelConsumptionBC = fuelConsumption \* distanceBC;

// Checking refueling amount

if (fuelConsumptionAB > MAX\_FUEL) {

flag == 1;

} else if (fuelConsumptionAB + fuelConsumptionBC > MAX\_FUEL) {

if(fuelConsumptionBC > MAX\_FUEL){

flag == 2;

}else{

refuelingAmount = MAX\_FUEL - fuelConsumptionAB;

flag == 3;

}

} else {

flag == 4;

}

//4. Output

switch (flag)

{

case 0:{

cout << "Overweight!!! The plane cannot lift." << endl;

exit(1);

}

break;

case 1:{

cout << "The aircraft cannot reach point B." << endl;

exit(1);

}

break;

case 2:{

cout << "The aircraft cannot fly from B to C." << endl;

}

break;

case 3:{

cout << "The aircraft needs to refuel " << refuelingAmount << " liters at point B." << endl;

}

break;

case 4:{

cout << "The aircraft can fly from A to C without refueling at B." << endl;

}

break;

default:

break;

}

return 0;

}