William Kelley

CS318-ASU

Week 8 Program 1

**Recursive Workout-SOURCE**

#include <iostream>

#include <iomanip>

#include <fstream>

#include <string>

using namespace std;

ofstream outfile;

ifstream infile;

const int size = 9;

int ary[size] = { 48, 54, 77, 92, 87, 65, 48, 21, 81 };

int \*p;

void arrayinfile(int \*array, int size);

int Digit\_Sum(int x);

int digit\_sum\_tosingle(int x);

int Sum\_Array\_Elements(int array[], int size, int sum);

int Sum\_insteadof\_Multiplication(int x, int y);

int sum\_from\_bottom(int x);

int DigCount(int x);

void int\_reverse(int x);

void print(int \*array, int size);

int main()

{

outfile.open("recursiveresults.txt");

int a, b, c, d, e, f, g, sum(0);

p = ary;

a = Sum\_Array\_Elements(ary, size, sum);

outfile << "Sum of Array Elements: " << a << endl;

d = Sum\_insteadof\_Multiplication(37, 24);

outfile << "37 added 24 times: " << d << endl;

f = sum\_from\_bottom(120);

outfile << "Sum from bottom: " << f << endl;

g = DigCount(12345);

outfile << "Digit Count for 12345: " << g << endl;

g = DigCount(28);

outfile << "Digit Count for 28: " << g << endl;

outfile << "12345 in reverse: ";

int\_reverse(12345); outfile << endl;

outfile << "28 in reverse: ";

int\_reverse(28); outfile << endl;

outfile << "Forward then Reverse Array: " << endl;

print(p, size);

outfile << endl;

outfile << "Digit Sum:\n";

g = Digit\_Sum(123456);

outfile << "123456: " << g << endl;

g = Digit\_Sum(90909);

outfile << "90909: " << g << endl;

g = Digit\_Sum(876543219);

outfile << "876543219: " << g << endl;

outfile << "Digits Sum to Single Digit:\n";

g = digit\_sum\_tosingle(9891234);

outfile << "9891234: " << g << endl;

g = digit\_sum\_tosingle(878989);

outfile << "878989: " << g << endl;

g = digit\_sum\_tosingle(999999);

outfile << "999999: " << g << endl;

outfile.close();

return 0;

}

void arrayinfile(int \*array, int size)

{

for (int i = 0; i < size; i++)

{

cin >> \*array;

array++;

}

}

//#4 from Book

int Sum\_Array\_Elements(int array[], int size, int sum) // must use int \*array instead of array[]

{

if (size < 0)

return sum;

else

sum += array[size];

return Sum\_Array\_Elements(array, --size, sum);

}

//#5 from Book

int Sum\_insteadof\_Multiplication(int x, int y)

{

int sum = y;

if (x == 1)

return sum;

else

return sum + Sum\_insteadof\_Multiplication(x - 1, y);

}

//#7 from Book

int sum\_from\_bottom(int x)

{

int sum = x; // may need to set sum = 0;

if (x == 0)

return sum;

else

return sum + sum\_from\_bottom(--x);

}

//#1 from Sheet

int DigCount(int x)

{

if (x <= 0)

return 0;

else

return 1 + DigCount(x / 10);

}

//#2 from Sheet

// use character array for input so it makes it easier to reverse

void int\_reverse(int x)

{

if (x == 0)

return;

else

{

outfile << x % 10;

return int\_reverse(x / 10);

}

}

//#3 from Sheet

void print(int \*array, int size)

{

if (size == 0)

{

return;

}

else

{

if (\*array != 0)

{

outfile << \*array << " ";

}

print(array+1, size-1);

if (\*array != 0)

{

outfile << \*array << " ";

}

return;

}

}

//#4 from Sheet

int Digit\_Sum(int x)

{

if (x == 0)

return 0;

else

return x % 10 + Digit\_Sum(x / 10);

}

//#5 from Sheet

int digit\_sum\_tosingle(int x)

{

if (x <= 9)

return x;

else

return digit\_sum\_tosingle(Digit\_Sum(x));

}