ECE118

First Test

1st October 2020

Name:

Student number:

Sign here if you did not give or receive aid in any form during this test, and did not consult any written or printed material apart from this test:

Don’t write in these boxes.

|  |  |  |
| --- | --- | --- |
| Question | Out of | Grade |
| 1 | 33 |    |
| 2 | 33 |  |
| 3 | 34 |  |

1.

Exactly what would be printed by this program?

Here’s a clue: the first line would be

a: 42 (int)

NO CALCULATORS

void show(const string question, const int n)

{ cout << question << ": " << n << " (int)\n"; }

void show(const string question, const double d)

{ cout << question << ": " << d << " (double)\n"; }

void show(const string question, const bool b)

{ cout << question << ": " << b << " (bool)\n"; }

void main()

{ show("a", 6 \* 7);

A: 42 (int)

 show("b", 2\*2\*2\*2 / 2\*2\*2\*2);

2\*2=4 \*2=8 \*2=16 /2 =8 \*2 = 16 \*2 = 32 \*2 = 64

B: 64 (int)

 show("c", 1+2 \* 3+4 \* 5+6);

1+2\*3+4\*5+6

1+6+4\*5+6

1+6+20+6

33

C: 33 (int)

 show("d", 10 / 3 \* 6);

d: 18 (int)

 show("e", 1 / 4 + 1 / 4 + 1 / 2);

1/4+1/4+1/2

0+0+0

E: 0 (int)

 show("f", 382757 % 2);

f: 1 (int)

 show("g", 879321729 % 1000);

g: 729 (int)

 show("h", (int)(1.24 \* 4));

1.24 \*

 4

4.96

 1

H: 4 (int)

 show("i", 382757 - 382757 / 1000 \* 1000);

382757 - 382757 / 1000 \* 1000

382757 - 382 \* 1000

382757 – 382000

I: 757 (int)

 show("j", 382757.0 - 382757 / 1000 \* 1000);

382757.0 - 382757 / 1000 \* 1000

382757.0 - 382 \* 1000

382757.0 - 382000

382757.0 – 382000.0

J: 757 (double)

 show("k", 382757 - 382757 / 1000.0 \* 1000);

382757 - 382757 / 1000.0 \* 1000

382757 – 382757.0 / 1000.0 \* 1000

382757 – 382.757 \* 1000

382757 – 382757.0

K: 0 (double)

 show("l", 382757 - 382757 / 1000 \* 1000.0);

382757 - 382757 / 1000 \* 1000.0

382757 - 382 \* 1000.0

382757 – 382000.0

L: 757 (double)

 show("m", 382757 - 382757.0 / 1000 \* 1000);

382757 - 382757.0 / 1000 \* 1000

382757 – 382.757 \* 1000

382757 – 382757.0

M: 0 (double)

 show("n", 4 / 3 > 1);

1 > 1

N: false (bool)

N: 0 (bool)

 show("o", 4 / 3 > 1.0);

4/3>1.0

1>1.0

1.0>1.0

O: false (bool)

 show("p", 15 / 2 / 2.0); }

15 / 2 / 2.0

7 / 2.0

P: 3.5 (double)

part q: get parts a to p right!

2. For this question, only consider positive numbers.

a. Analyse this function

void aaa(int x)

{ if (x == 0)

 new\_line();

 else

 { print("+");

 aaa(x - 1); } }

State in plain English what the function does - its overall effect, not a description of the individual statements, and show how you arrived at your conclusion.

What does aaa(0) do? Prints \n

What does aaa(1) do? Prints + then does aaa(0)

 Prints + then prints \n

What does aaa(2) do? Print + then aaa(1)

 = print +\n

 Print ++\n

What does aaa(3) do? Print + then aaa(2)

 = prints ++\n

 Prints +++\n

Aaa(x) prints x plusses followed by a new line.

e.g. aaa(8) prints ++++++++\n

b. Now do the same for this function

void bbb(int x)

{ if (x == 0)

 { print("\*");

 new\_line(); }

 else

 { bbb(x - 1);

 aaa(x); } }

bbb(0) prints \*\n

what does bbb(1) do? Bbb(0) followed by aaa(1)

 print \*\n then print +\n

\*

+

What does bbb(2) do? Bbb(1) followed by aaa(2)

\*

+

++

What does bbb(3) do? BBB(2) followed by aaa(3)

\*

+

++

+++

What does bbb(4) do? Bbb(3) followed by aaa(4)

\*

+

++

+++

++++

Bbb(8)

\*

+

++

+++

++++

+++++

++++++

+++++++

++++++++

c. What does this function compute?

int ccc(int x)

{ if (x <= 1)

 return 1;

 else

 { const int a = ccc(x - 1);

 const int b = ccc(x - 1);

 return a + b + 1; } }

 what is the value of ccc(2)?

 what is the value of ccc(4)?

 what is the value of ccc(9)? Remember to show your work.

Ccc(1) = 1

Ccc(2) =

 A = ccc(1) = 1

 B = ccc(1) = 1

 Answer is a + b + 1

= 3

Ccc(3) =

 A = ccc(2) = 3

 B = ccc(2) = 3

 Answer is a + b + 1

 = 7

Observer a and b will always have the same value

So

Ccc(4): a = b = 7, answer 7+7+1 = 15

Ccc(5): a = b = 15, answer a+b+1 = 31

Ccc(6): a = b = 31, answer is a+b+1 = 63

Example: CCC(19) = 2\*ccc(18) + 1

Ccc(7) = 2 \* ccc(6) + 1 = 2\*63+1 = 127

Ccc(8) = 2 \* ccc(7) + 1 = 2\*127 + 1 = 255

Ccc(9) = 2\*ccc(8)+1 = 2\*255 + 1 = 511

1 3 7 15 31 63 127 255 511

Ccc(x) = two to the power of x minus 1

3. Use only constants, no variables in your answers to this question.

 Use recursive functions to cause repetition when needed.

a. Write a function PrintArray(R, x, y), where R is an array of ints. It should print all the ints from position x to position y in the array.

e.g. const int b[] = { 7, 3, 2, 6, 15, 3, 9, 5 };

 PrintArray(b, 2, 5);

prints 2 6 15 3.

Void printarray(int R[], int x, int y)

{ if (x > y)

 Return;

 Cout << R[x] << “ “;

 Printarray(R, x + 1, y); }

Void printarray(int array[], int x, int y)

{ if (x == y)

 Cout << array[x] << “\n”;

 else

 { Cout << array[x] << “ “;

 Printarray(array, x + 1, y); } }

b. Write another function AvgArray(R, n), where R is an array of ints, and n is the size of that array. It should return as its result the average of all the ints in the array.

e.g. cout << AvgArray(b, 8)

prints 6.25 because 7+3+2+6+15+3+9+5 = 50

int AddUpArray(int R[ ], int n)

{ if (n == 0)

 Return 0;

 Return AddUpArray(R, n-1) + R[n-1]; }

double AvgArray(int R[ ], int n)

{ return AddUpArray(R, n) / (double)n; }

X = AddUpArray(b, 4)

 = (AddUpArray(b, 3)) + b[3]

 = ((AddUpArray(b, 2)) + b[2]) + b[3]

 = (((AddupArray(b, 1)) + b[1]) + b[2]) + b[3]

 = (((AddUpArray(b, 0) + b[0]) + b[1]) + b[2]) + b[3]

 = (((0 + b[0]) + b[1]) + b[2]) + b[3]

How do I add up all N elements of R?

 Add up the first N-1 elements of R, then add R[N-1] to the result

 Except if N is 0, the answer is 0

c. Write a function RangeArray(R, n), where R is an array of ints, and n is the size of that array. It should return as its result the difference between the largest and the smallest ints in the array

e.g. RangeArray(b, 8) is 13, because that is 15 – 2

int min(int a, int b)

{ if (a < b)

 Return a;

 Return b; }

Same sort of thing for max

int largest(int R[ ], int n)

{ if (n == 1)

 Return R[0];

 Return max(largest(R, n-1), R[n-1]); }

int smallest(int R[ ], int n)

{ if (n == 1)

 Return R[0];

 Return min(smallest(R, n-1), R[n-1]); }

Int rangearray(int R[ ], int n)

{ largest(R, n) – smallest(R, n); }