CS1010

https://t.me/cs1010isfun

Today's plan

- Tutorial Segment
  - *Unix & Clang*
  - *Recap on C*
  - *Discussion of problem sets*
- Lab Segment
  - *Programming Assignment (PA0)*

evantay@comp.nus.edu.sg

KAHOOT!

Quick quiz

Unix commands

- https://nus-cs1010.github.io/1819-s1/unix/index.html
- http://cheatsheetworld.com/programming/unix-linux-cheat-sheet/

evantay@comp.nus.edu.sg

Compiling with Clang

clang -Wall -g teh.c -o teh –lm

- *What is –Wall?*
- *What is –g?*
- *What is –o teh?*
- *What is –lm?*

evantay@comp.nus.edu.sg

Compiling with Clang

clang -Wall -g teh.c -o teh –lm

- *What is –Wall? Enable **W**arnings **All***
- *What is –g? **G**enerate additional info for lldb debugger*
- *What is –o teh? **O**utput file name set to teh*
- *What is –lm? **L**ink to **M**ath library*

https://nus-cs1010.github.io/1819-s1/clang/index.html

evantay@comp.nus.edu.sg

Today's plan

- Tutorial Segment
  - *Unix & Clang*
  - ***Recap on C***
  - *Discussion of problem sets*
- Lab Segment
  - *Programming Assignment (PA0)*

evantay@comp.nus.edu.sg

Recap on C

```c
#include <math.h>
#include "cs1010.h"

long square(long x)
{
  return x*x;
}

double hypotenuse_of(long base, long height)
{
  return sqrt(square(base) + square(height));
}

int main()
{
  double hypotenuse;
  long base = cs1010_read_long();
  long height = cs1010_read_long();
  hypotenuse = hypotenuse_of(base, height);
  cs1010_println_double(hypotenuse);
}
```

Today's plan

- Tutorial Segment
  - *Unix & Clang*
  - *Recap on C*
  - *Discussion of problem sets*
- Lab Segment
  - *Programming Assignment (PA0)*

evantay@comp.nus.edu.sg

PROBLEM SETS

3.1, 3.2, 5.1, 5.2

Problem Set 3.1

$$\frac{\sum_{i=0}^{k-1} |l_i - \mu|}{k}$$

- MAD: How spread out a set of data is.
- Absolute deviation: Absolute difference between an element and the mean.
- MAD: Mean of all absolute deviations.

evantay@comp.nus.edu.sg

Problem Set 3.1

$$\frac{\sum_{i=0}^{k-1} |l_i - \mu|}{k}$$

- Given the following functions: mean(L, k), subtract(L, k, s), abs(L, k), how do you solve this problem?
  - *L: List of elements; k: Number of elements in L*
  - *mean(L, k): Returns mean of list L*
  - *subtract(L, k, s): Subtracts s from every element in L*
  - *abs(L, k): Absolutes every element in L*

evantay@comp.nus.edu.sg

Problem Set 3.1

$$\frac{\sum_{i=0}^{k-1} |l_i - \mu|}{k}$$

- Find the mean:
  - *mean(L, k)*
- Subtract the mean from every element
  - *subtract(L, k, mean(L, k))*

evantay@comp.nus.edu.sg

Problem Set 3.1

$$\frac{\sum_{i=0}^{k-1} |l_i - \mu|}{k}$$

- Absolute every element in the list:
  - *abs(subtract(L, k, mean(L, k)), k)*
- Find the mean again
  - *mean(abs(subtract(L, k, mean(L, k)), k), k)*

evantay@comp.nus.edu.sg

PROBLEM SETS

3.1, **3.2**, 5.1, 5.2

Problem Set 3.2 (a)

Find the sum of all the integers in the list L with k integers (k>0) that is recursive.

```
sum(L, i, j) {
        if ( i == j ) {
                return Li;
        } else {
                return Li + sum(L, i+1, j);
        }
}
```


evantay@comp.nus.edu.sg

Problem Set 3.2 (b)

The function pow(i,j) computes i^j. How to compute recursively.

- pow(2,3) returns 2^3 = 2 x 2 x 2 = 8

evantay@comp.nus.edu.sg

sunfire-r.comp.nus.edu.sg - PuTTY

```c
#include "cs1010.h"

int recursive_pow(int integer, int exponent)
{
  if (integer == 1)
  {
    return integer;
  }
  else
  {
    return integer * recursive_pow(integer, --exponent);
  }
}

int main()
{
  cs1010_print_string("recursive_pow(2, 3) aka 2^3: ");
  int output = recursive_pow(2, 3);
  // First run of recursive_pow(2, 3) returns 2 * recursive_pow(2, 2);
  // Second run of recursive_pow(2, 2) returns 2 * recursive_pow(2, 1);
  // Third run of recursive_pow(2,1) returns 2;
  // Second run of recursive(pow(2, 2) returns 2 * 2;
  // First run of recursive_pow(2, 3) returns 2 * (2 * 2);
  cs1010_println_long(output);
}
```

1,1 All

Problem Set 3.2 (b)

The function pow(i,j) computes i^j. How to compute recursively.

PROBLEM SETS

3.1, 3.2(b), **5.1**, 5.2

Problem Set 5.1

Would passing an int into **sqrt(...)** result in an error?

```
double sqrt(double x);
```

```
double
hypotenuse_of(long base, long height)
{
  return sqrt(square(base) + square(height));
}
```


evantay@comp.nus.edu.sg

Problem Set 5.1

No, because any double can hold any int.

```
double sqrt(double x);
```

```
double
hypotenuse_of(long base, long height)
{
  return sqrt(square(base) + square(height));
}
```


evantay@comp.nus.edu.sg

sunfire-r.comp.nus.edu.sg - PuTTY

```
#include "cs1010.h"

int main() {
  cs1010_println_string("Double can hold any integer value without data loss");
  int i = 23;
  cs1010_print_string("Printing int i = 23 as double: ");
  cs1010_println_double(i);

  cs1010_println_string("Converting double to integer results in data loss");
  double d = 3.142;
  cs1010_print_string("Printing double d = 3.142 as integer: ");
  cs1010_println_long(d);
}
~
~
~
~
~
~
~
~
~
~
~
~
"problem5_1.c" 13L, 408C                    1,1           All
```

PROBLEM SET 5.1

sunfire-r.comp.nus.edu.sg - PuTTY

```
evantay@pe118:~/code/tut2$ ./problem5_1
Double can hold any integer value without data loss
Printing int i = 23 as double: 23.0000
Converting double to integer results in data loss
Printing double d = 3.142 as integer: 3
evantay@pe118:~/code/tut2$
```

PROBLEM SET 5.1

PROBLEM SETS

3.1, 3.2(b), 5.1, 5.2

Problem Set 5.2

What can the return type of this square be, in order for the return type to be big enough to store all possible values of x * x?

```
??? square(uint16_t x) {
    return x*x;
}
```

- Max value of input is 2^16 – 1 = 65,535
  - *(limited by **uint16_t** parameter)*

evantay@comp.nus.edu.sg

Problem Set 5.2

What can the return type of this square be, in order for the return type to be big enough to store all possible values of x * x?

```
??? square(uint16_t x) {
    return x*x;
}
```

- Max value of input is 2^16 – 1 = 65,535
  - *(limited by **uint16_t** parameter)*
- Can we use uint16_t for return type?

evantay@comp.nus.edu.sg

Problem Set 5.2

What can the return type of this square be, in order for the return type to be big enough to store all possible values of x * x?

```
??? square(uint16_t x) {
    return x*x;
}
```

- Max value of input is 2^16 – 1 = 65,535
  - *(limited by **uint16_t** parameter)*
- Can we use uint16_t for return type? No

evantay@comp.nus.edu.sg

Problem Set 5.2

What can the return type of this square be, in order for the return type to be big enough to store all possible values of x * x?

```
??? square(uint16_t x) {
    return x*x;
}
```

- Max value of input is 2^16 – 1 = 65,535
  - *(limited by **uint16_t** parameter)*
- Can we use uint32_t for return type?

evantay@comp.nus.edu.sg

Problem Set 5.2

What can the return type of this square be, in order for the return type to be big enough to store all possible values of x * x?

```
??? square(uint16_t x) {
    return x*x;
}
```

- Max value of input is 2^16 – 1 = 65,535
  - *(limited by **uint16_t** parameter)*
- Can we use uint32_t for return type? Yes
  - *(2 ^ 32) – 1 > ((2 ^ 16) – 1) ^ 2*

evantay@comp.nus.edu.sg

Problem Set 5.2

What can the return type of this square be, in order for the return type to be big enough to store all possible values of x * x?

```
??? square(uint16_t x) {
    return x*x;
}
```

- Max value of input is 2^16 – 1 = 65,535
  - *(limited by **uint16_t** parameter)*
- Can we use int32_t for return type?

evantay@comp.nus.edu.sg

Problem Set 5.2

What can the return type of this square be, in order for the return type to be big enough to store all possible values of x * x?

```
??? square(uint16_t x) {
    return x*x;
}
```

- Max value of input is 2^16 – 1 = 65,535
  - *(limited by **uint16_t** parameter)*
- Can we use int32_t for return type? **No**
  - *Range is from – (2 ^ 16) to (2 ^ 16) - 1*

evantay@comp.nus.edu.sg

Today's plan

- Tutorial Segment
  - *Unix & Clang*
  - *Recap on C*
  - *Discussion of problem sets*
- Lab Segment
  - ***Exercise 1: Freezer.c***

evantay@comp.nus.edu.sg

Q & A

THE END

https://t.me/cs1010isfun