DATA STRUCTURES
CS-UH 1050, SPRING 2022

Introduction & Basics to C++

LECTURE OUTLINES

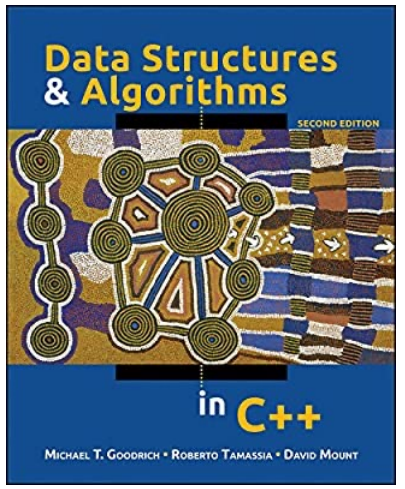
I. Course Logistics
II. Basics of C++

COURSE LOGISTICS

HELLO!

- Who am I?
  - Prof. Mai Oudah
  - mai.oudah@nyu.edu
  - Office hours: By Appointment
- Who is your lab and teaching assistant?
  - Mr. Khalid Mengal (kqm1@nyu.edu)
  - Office hours: By Appointment

COURSE DESCRIPTION



- Textbook:
  - Data Structures and Algorithms in C++ by Goodrich
- Main Objective:
  - You will learn how to utilize data structures for representing information in computer memory
  - Topics include: arrays, recursion, analysis of algorithms, lists, stacks, queues, trees, hashing, priority queues, dictionaries, graph data structures etc.
  - These data structures are to be implemented via C++ in this course

ASSESSMENT TASKS

- The components of the Final Grade include:

Quizzes (tentative days in course schedule)	15%
Midterm Exam (fixed date in the syllabus)	25%
Final Exam (TBD on Albert)	30%
Course Assignments	20%
Lab exercises (weekly)	10%

ASSIGNMENT SUBMISSION POLICY

- All course assignments are due at 11:59pm on the due date Abu Dhabi time.
- Submission MUST be done through NYU LMS (Brightspace).
- You have up to 2 late days to submit an assignment
  - For late submissions, 5% will be deducted from the homework grade per each late day.
- You MUST follow the code of conduct provided with every assignment or lab exercise and adhere to the standards of academic integrity at NYUAD.

ASSIGNMENT EXTENSION PROTOCOL

- Extensions will NOT be granted on or after the deadline
- Applications for extension should be filed as early as possible and before the assignment due date by at least 3 working days (weekend days are not counted)
- Any application for assignment extension MUST include supporting documentation, such as a medical certificate and/or other official evidence

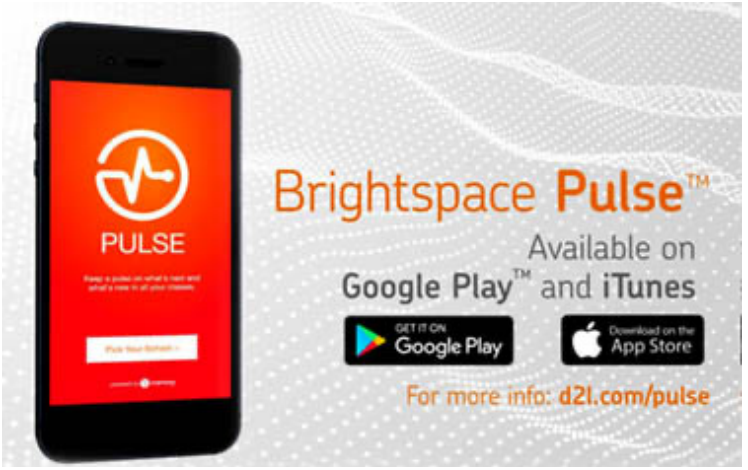
GRADES TO LETTERS MAPPING

No grading on a curve here

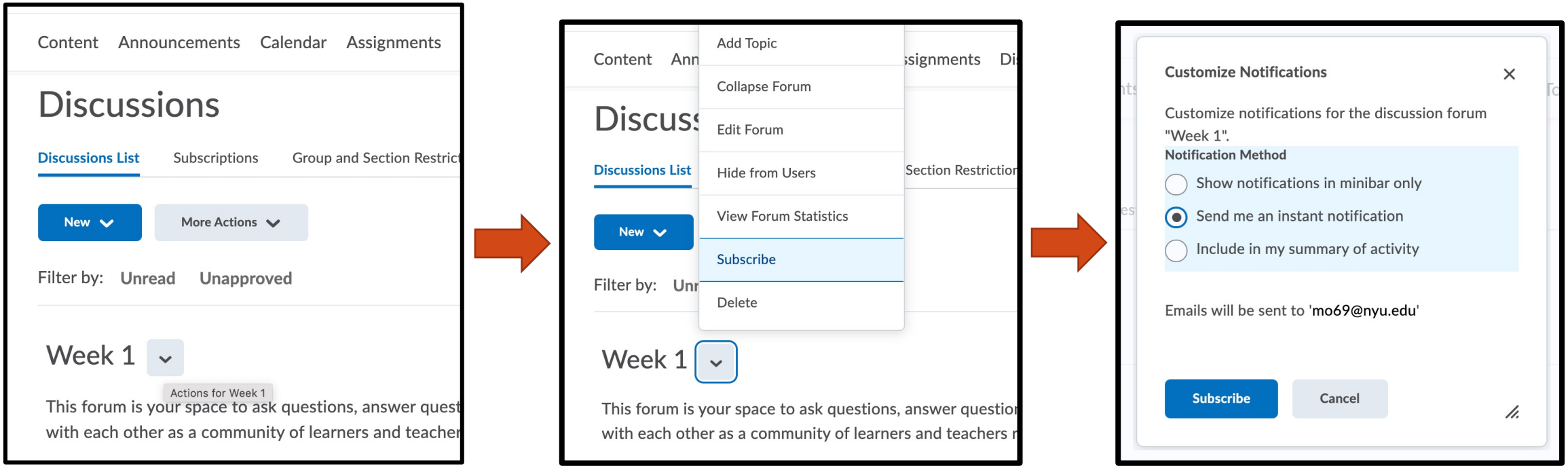
Minimum Score	Grade
[95-100]	A
[90-95[	A-
[87-90[	B+
[83-87[	B
[80-83[	B-
[77-80[	C+
[73-77[	C
[70-73[	C-
[67-70[	D+
[63-67[	D
[0-63[	FAIL

NYU BRIGHTSPACE

- You can find the course logistics, syllabus and all other material on the course site.
- Don’t forget to subscribe in the weekly discussions!
- Access via https://brightspace.nyu.edu/
- There is a mobile app for Brightspace users that can help you stay connected and on track with your courses: Brightspace Pulse



SUBSCRIBE IN DISCUSSIONS



CAMERA USAGE DURING CLASS

- Students are expected to participate in synchronous classes with their video cameras turned on and with their name registered on Zoom.
- Exceptions to this expectation should be discussed with the faculty member or course instructor. Our Academic Technology team is also available for support to address any connectivity issues students may have.
- There will be no recording of the zoom sessions.

BASICS OF C++

C++: INTRO

- Semicolon (;) at the end of each line of executable code
- How to write comments:
  - On one or more lines

```
/*
  Text on one or more lines
*/
```

  - On a single line

```
// text
```

C++: DATA TYPES

- **Main Primitive Data Types:** (built-in or predefined)
  - **int** = Integer e.g. `int x = 3; //if the value is 3.5, then x = 3`
  - **char** = Character e.g. `char r = 'g'; //char s[256] = "hey";`
  - **bool** = Boolean e.g. `bool flag = true;`
  - **float** = 32-bit floating-point number e.g. `float x = 3.5;`
  - **double** = 64-bit floating-point number e.g. `double y = 5.6;`
  - **void** = without any value. Void data type is used for those function which does not returns a value. e.g. `void fun(int x, int y)`

C++: DATA TYPES

- **Strings:** To use strings, you must include a *header* in the source code; the **<string>** library:

```
#include <string>
string s = "to be";
int i = s.size();  // now i equals 5
string t = "not " + s;  // t = “not to be”
string u = s + " or " + t;  // u = “to be or not to be”
```

C++: DATA TYPES

- **Arrays:** To declare an array: 1) define the variable data type, 2) specify the array's name followed by **[ ]**, and 3) specify the number of elements in the array:

Example:

```
int x[3]; // here, x is defined as an array of 3 integers
x[0]=20; x[1]=40; x[2]=60;
```

The above statements can be done in one go as follows:

```
int x[] = {20, 40, 60}; // x is an array of 3 integers
```

C++: DATA TYPES

- **Constants:** make a declared variable unchangeable and read-only

Example:

```
const int num = 15;  // num will always be 15
```

C++: DATA TYPES

- **Enumeration:** An enumeration is a user-defined data type that consists of a fixed set of constants

E.g.
```
enum  Card  {club = 0, diamonds = 10, hearts = 20, spades = 3};
    Card  x;  // Now you can define a variable, x, of type Card
    x = spades;  // the value of x is now 3
```

If you do not give values to elements, they will automatically be assigned: 0, 1, 2, 3, 4, …

E.g.
```
enum  Day  {SUN, MON, TUE, WED, THU, FRI, SAT};
    Day x = SUN;  // the value of x is now 0
```

C++: OPERATORS

- Comparison Operators:

  `==, !=, <, >, <=, >=`

- Arithmetic Operators:

  `+, -, *, /, %`

- Logic Operators:

  `&&` means logical AND

  `||` means logical OR

  `!` means logical NOT

- Increment/decrement Operators:

  `i++` means use `i`, then increase `i` by 1

  `++i` means increase `i` by 1, then use it

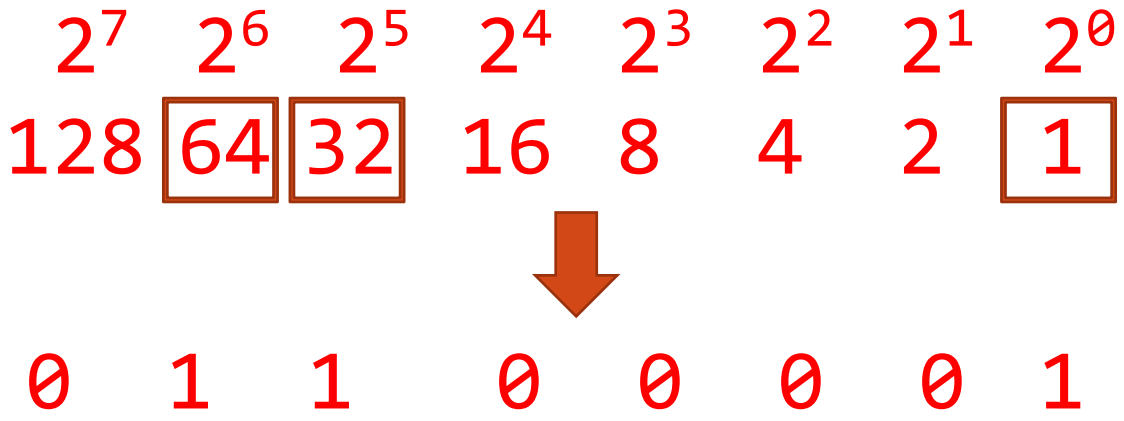
Example:

```
int a[] = {0, 1, 2, 3};
int i = 2;
int j = i++;      // j=2 and now i=3
int k = --i;      // now i=2 and k=2
cout<< a[k++];    // output 2, and now k=3

i += 5;  // now i=7
k -= 2;  // now k=1
```

C++: BINARY REPRESENTATION

- Suppose that k = ‘a’, the binary representation of its ascii code (i.e. 97) can be determined as follows:



C++: BINARY REPRESENTATION

- Subset of the ASCII table for your reference:

Character	ASCII
a	97
b	98
c	99
d	100
e	101
f	102
g	103
h	104
i	105
j	106
k	107
l	108
m	109

Character	ASCII
n	110
o	111
p	112
q	113
r	114
s	115
t	116
u	117
v	118
w	119
x	120
y	121
z	122

Character	ASCII
A	65
B	66
C	67
D	68
E	69
F	70
G	71
H	72
I	73
J	74
K	75
L	76
M	77

Character	ASCII
N	78
O	79
P	80
Q	81
R	82
S	83
T	84
U	85
V	86
W	87
X	88
Y	89
Z	90

Character	ASCII
0	48
1	49
2	50
3	51
4	52
5	53
6	54
7	55
8	56
9	57

C++: BITWISE OPERATORS

- Suppose that the bitwise representation of a variable, `a`, is `00000101`, and another variable, `b`, is `00001001`

Bitwise operator:		Example:
`~exp`	bitwise **complement**	`~a  // 11111010`
`exp & exp`	bitwise **and**	`a&b  // 00000001`
`exp ^ exp`	bitwise **exclusive-or (XOR)**	`a^b  // 00001100`
`exp | exp`	bitwise **or**	`a|b  // 00001101`
`exp1 << exp2`	**shift** exp1 **left** by exp2 bits	`b<<1 // 00010010`
`exp1 >> exp2`	**shift** exp1 **right** by exp2 bits	`b>>1 // 00000100`

The **left-shift operator** fills with zeros.
The **right-shift operator** fills with zeros if `exp1` is an unsigned variable. Otherwise, it fills with 0 if `exp1` is positive, and with 1 if `exp1` is negative.