

# TEMPLATES

---

Problem Solving with Computers-I

<https://ucsb-cs24-sp17.github.io/>

C++

```
#include <iostream>
using namespace std;

int main(){
    cout<<"Hola Facebook\n";
    return 0;
}
```



# How is PA3 going?

- A. Done!
- B. Done with part 1. On-track to finish part2
- C. Half way through both part 1 and part 2
- D. Long way to go
- E. Haven't started

# Announcements

- PA3 is due today (5/8)
- PA4 is due in a week (5/15)
- PA4 must be done individually

# Polynomial class

- Differences between PA3 and PA4

# Finding the Maximum of Two Integers

Here's a small function that you might write to find the maximum of two integers.

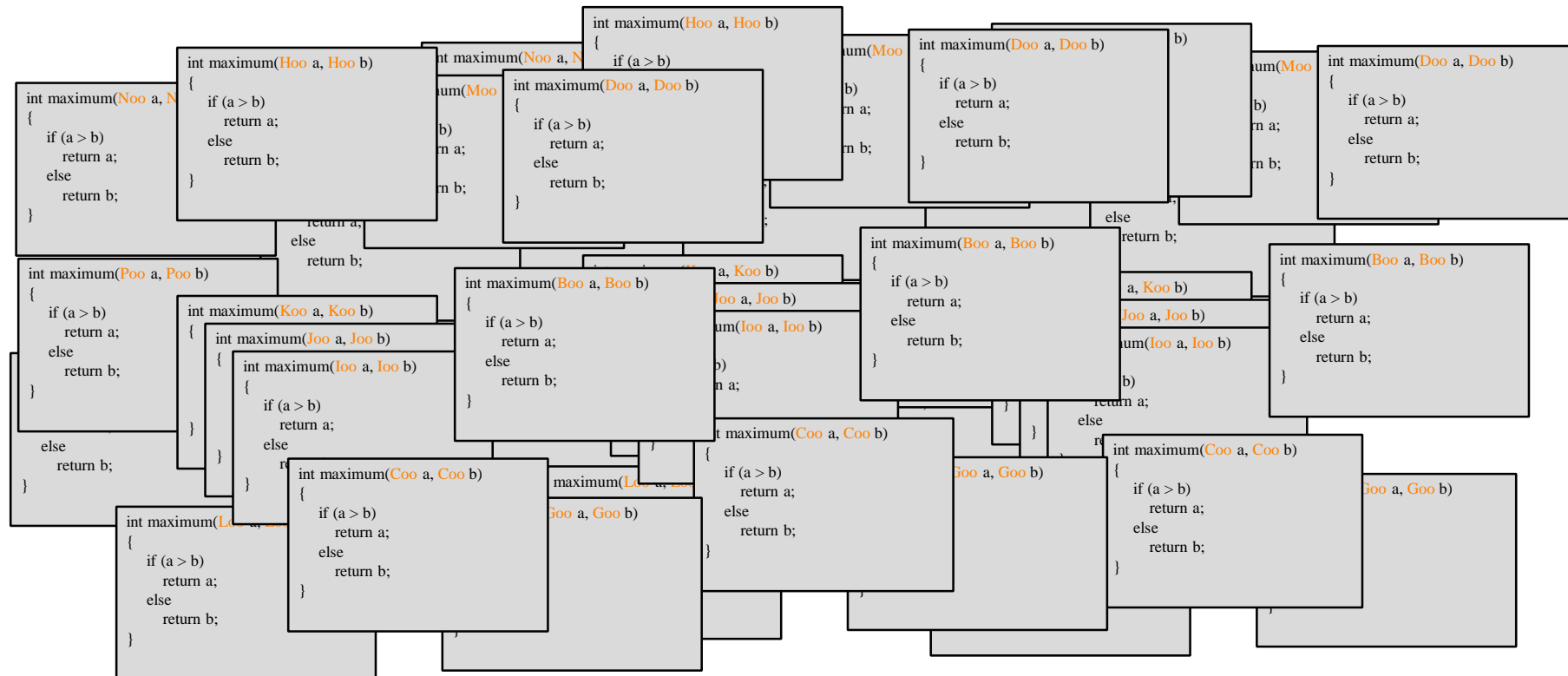
```
int maximum(int a, int b)
{
    if (a > b)
        return a;
    else
        return b;
}
```

# Finding the Maximum of Two Points

```
Point maximum(Point a, Point b)
{
    if (a > b)
        return a;
    else
        return b;
}
```

# One Hundred Million Functions...

- Suppose your program uses 100,000,000 different data types, and you need a maximum function for each...



# A Template Function for Maximum

- When you write a template function, you choose a data type for the function to depend upon...

```
template <class Item>
Item maximum(Item a, Item b)
{
    if (a > b)
        return a;
    else
        return b;
}
```



# What are the advantages over typedef?

```
template <class Item>
Item maximum(Item a, Item b)
{
    if (a > b)
        return a;
    else
        return b;
}
```

```
typedef int item;
item maximum(item a, item b)
{
    if (a > b)
        return a;
    else
        return b;
}
```

# A Template Function for Maximum

Is the following a valid template function?

- A. Yes
- B. No

```
template <class Item>
Item maximum(int a, int b)
{
    Item result;
    if (a > b)
        result = a;
    else
        result = b;
    return result;
}
```

# Template classes

## Using a Typedef Statement:

```
class bag
{
  public:
    typedef int value_type;
    . . .
```

## Using a Template Class:

```
template <class Item>
class bag
{
  public:
    typedef Item value_type;
    . . .
```

# Template classes: Non-member functions

*bag operator +(const bag& b1, const bag& b2)...*

```
template <class Item>
```

```
bag<Item> operator +(const bag<Item>& b1, const bag<Item>& b2)...
```

# Template classes: Member function prototype

- Rewrite the prototype of the member function “count” using templates

Before (without templates)

```
class bag{
    public:
        typedef std::size_t size_type;
        ....
        size_type count(const value_type& target) const;
        .....
};
```

# Template classes: Member function definition

```
bag::size_type bag::count(const value_type& target) const ...
```

The function's return type is specified as `bag::size_type`. But this return type is specified before the compiler realizes that this is a `bag` member function. So we must put the keyword *typename* before `bag<Item>::size_type`. We also use `Item` instead of `value_type`:

```
template <class Item>
typename bag<Item>::size_type bag<Item>::count
    (const Item & target) const ...
```

# Template classes: Including the implementation

```
#include "bag4.template" // Include the implementation.
```

## How to Convert a Container Class to a Template

1. The template prefix precedes each function prototype or implementation.
2. Outside the class definition, place the word `<Item>` with the class name, such as `bag<Item>`.
3. Use the name `Item` instead of `value_type`.
4. Outside of member functions and the class definition itself, add the keyword *typename* before any use of one of the class's type names. For example:  

```
typename bag<Item>::size_type
```
5. The implementation file name now ends with `.template` (instead of `.cxx`), and it is included in the header by an include directive.
6. Eliminate any using directives in the implementation file. Therefore, we must then write `std::` in front of any Standard Library function such as `std::copy`.
7. Some compilers require any default argument to be in both the prototype and the function implementation.



# Using a template class

```
bag<string> adjectives; // Contains adjectives typed by user  
bag<int>    ages;      // Contains ages in the teens  
bag<string> names;    // Contains names typed by user
```