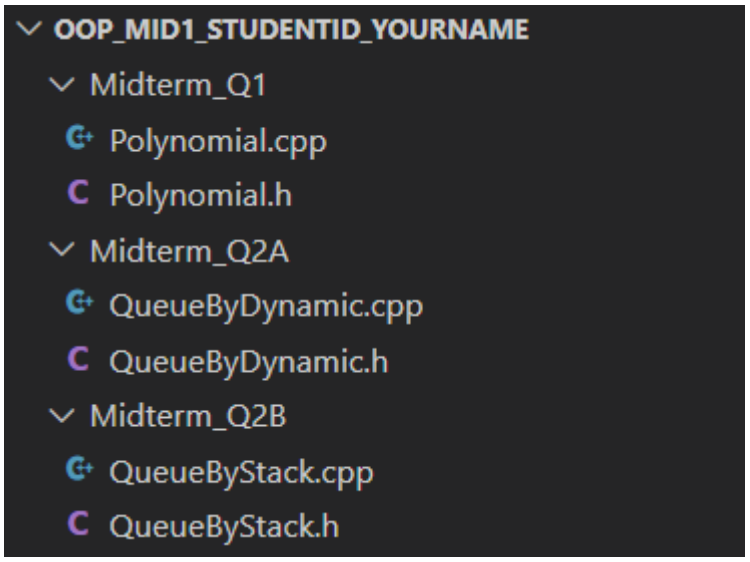
OOP - Midterm 1 - 2024.04.11

Exam Time: 18:30 ~ 21:20
There are two questions. You must finish all the questions.

Rules

1. Put your student ID card on the table.
2. If you want to go to the toilet, raise your hand and tell TA.
3. The main.cpp file for each question:
   It is only a basic test for your code. We will also test your code with other test data.
4. You can start to upload your answer files after **20:45** and before the deadline.
5. TA will not accept any reason for uploading files too late except e3 is crashed.
6. Turn in:
   - Make your directory as below



   - There should be your answer file (.cpp, .h) in each directory.
   - You don't need to submit main_Q1.cpp, main_Q2A.cpp, main_Q2B.cpp.
   - Zip all files in one file and name it OOP_mid1_studentID_YourName.zip
   - If you don't follow this rule, your score will be reduced by 30 points.
   - If you submit the wrong code (ex: empty .cpp file), there is no second chance this time and your score is zero.
7. You must use the template to do this midterm.
8. You are not allowed to include other libraries except for the ones specified in the question.
9. You must follow the output format.
10. If your source code cannot be built, your score is 0.
11. If you cheat, your score is 0.

Q1. Polynomial (50%)

In this question, you need to create a polynomial class to perform some basic calculations related to differentiation.

Do not change the names of the functions and the class.
Please complete the functions in public.
You must use **protected member variables** to implement this class. (i.e. Access to member variables from external sources is restricted.)
Do not call any inbuilt data structure (like std::vector)

Assume all of the coefficients and exponents are **integers**. $(0 \leq exponents \leq 20)$
The operations are listed below:

Operations	**Description**
A. (5%) addTerm(coef, exp)	Add the term $coef * x^{exp}$ to the polynomial.
B. (5%) setCoef(exp, coef)	Set the coefficient of the specified exponent term with **exp** to **coef**. If the exponent does not exist, add it as a new term.
C. (5%) getCoef(exp)	Return the coefficient of the term with the exponent **exp**. If the exponent does not exist, return 0.
D. (5%) Polynomial(poly) operator =(poly)	Implement the copy constructor. You may need this function for the functions below. Hint: When implementing the operator+(), it should return a new instance. For example, if A = B + C, A will not be the same instance as B, but rather a new one.
E. (5%) operator +(poly)	Add another polynomial instance to this instance. E.g. $(2x^3 - 1) + (x^2 + 4x) = 2x^3 + x^2 + 4x - 1$
F. (5%) operator -(poly)	Subtract this instance by another polynomial instance. E.g. $(2x^3 - 1) - (x^2 + 4x) = 2x^3 - x^2 - 4x - 1$
G. (5%) operator *(poly)	Multiply this instance by another polynomial instance. E.g. $(2x^3 - 1) * (x^2 + 4x) = 2x^5 + 8x^4 - x^2 - 4x$
H. (5%) differential()	Do the differential on this instance and return it. E.g. $d(x^3 - 2x + 1)/dx = 3x^2 + 2$ Note: $d(cx^n)/dx = cx^{n-1}$, where c and n are constants.
I. (5%) solve(x)	Evaluate the polynomial with a certain value of x. E.g. $f(x) = x^3 - 2x + 1,\ f(2) = 5$
J. (5%) print()	Print the polynomial. E.g. $3x^2 - 5x + 10$, you need to output **3x^2 - 5x + 10**

Input:

Use main_Q1.cpp to test your code.
Don't modify the content of main_Q1.cpp.

Output:

This is the result of running the main_Q1.cpp.
You should follow the output format.

```
========== Example 1 ==========
poly1 = 1
poly1 = x^2 + 1
poly1 = 3x^3 + x^2 + 1
poly1 = 3x^3 + 3x^2 + 1

poly1 = -3x^3 + 3x^2 + 1
The coefficient of the term with exponent 3 in poly1 is -3

========== Example 2 ==========
poly2 = 2x^3 - 1
poly3 = x^2 + 4x

poly2 + poly3 = 2x^3 + x^2 + 4x - 1
poly2 - poly3 = 2x^3 - x^2 - 4x - 1
poly2 * poly3 = 2x^5 + 8x^4 - x^2 - 4x

========== Example 3 ==========
poly5 = x^3 - 2x + 1

d(poly5)/dx = 3x^2 - 2
f(x) = poly5, f(2) = 5
```

Q2. Queue (50%)

This question has two parts. In the first part, you have to implement a queue based on a dynamic array. In the second part, you will utilize the stack in the standard library to create a queue.

- ## First part: Dynamic Array

You must use **dynamic memory** to implement this class, **or you'll get 0 score**.
Do not change the names of the functions and the class.
Please complete the functions in public.
Please implement the queue **BY YOURSELF**.
Do not call any inbuilt data structure (like std::queue, std::vector)

Assume all the input data are **positive integers**.
The operations are listed below:

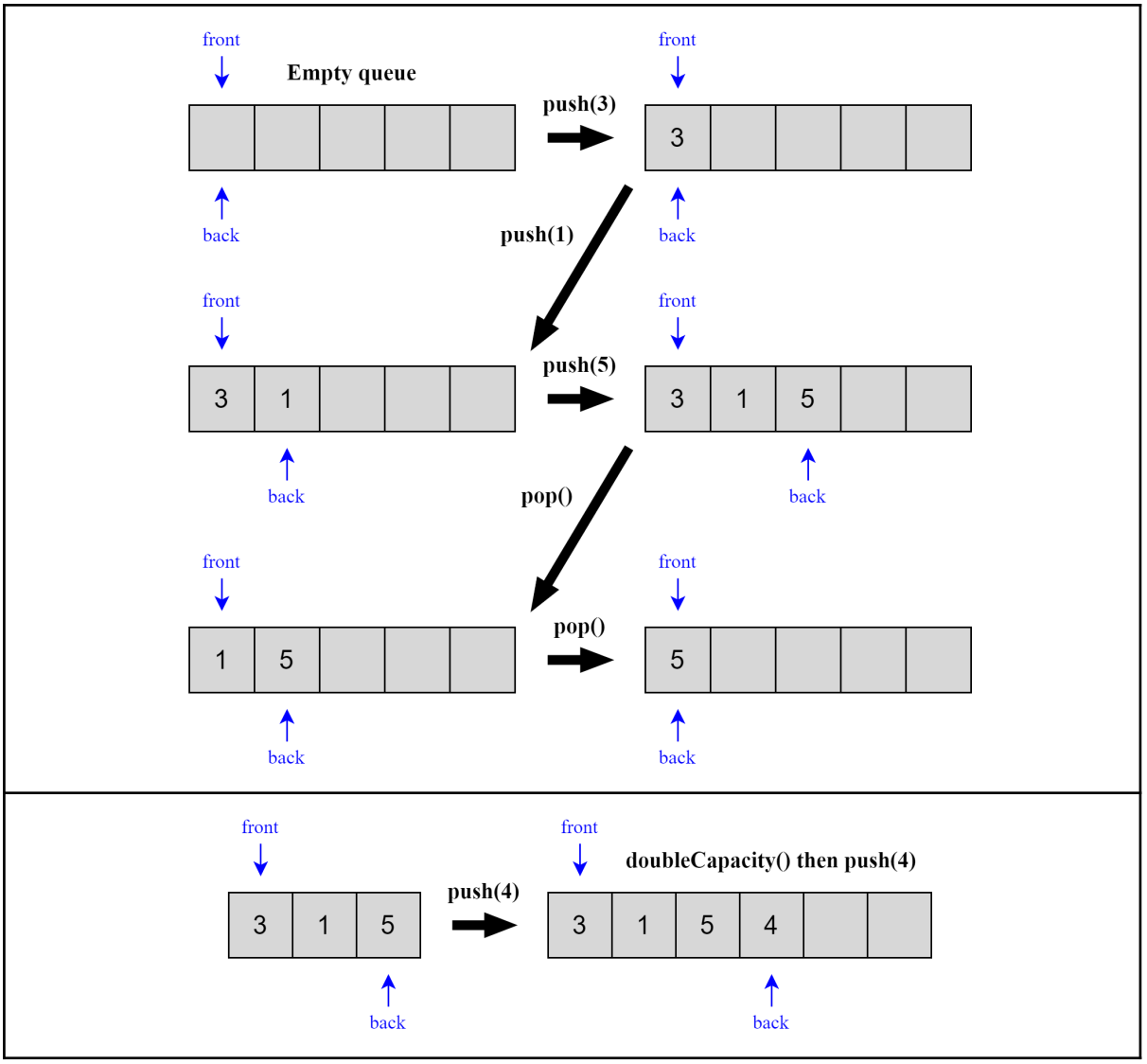
Operations	**Description**
A. (5%) QueueByDynamic q(capacity)	Class constructor. Create a dynamic array with an initial size of capacity.

B. (5%) push(x)	Add an element **x** at the end of the queue. If the queue is full, double the capacity of the queue first, and then push the element **x**.
C. (5%) pop()	Remove an element at the head of the queue. If there is no element in the queue, do nothing.
D. (5%) front()	Return the value at the head of the queue. If there is no element in the queue, return -1.
E. (5%) back()	Return the value at the tail of the queue. If there is no element in the queue, return -1.
F. (5%) doubleCapacity()	Double the capacity of the queue.

The following operations have been implemented for you, please **do not modify** it.

Operations	**Description**
G. (0%) size()	Return the current size of the queue.
H. (0%) capacity()	Return the capacity of the queue. The maximum number of elements that can be stored.
I. (0%) empty()	Return true if the queue is empty. Otherwise return false.
J. (0%) full()	Return true if the queue is full. Otherwise return false.
K. (0%) print()	Print the values of the elements in the queue from front to back.

Example:



Input:

Use main_Q2A.cpp to test your code.
Don't modify the content of main_Q2A.cpp.

Output:

This is the result of running the main_Q2A.cpp.
You should follow the output format.

```
========== Example 1 ==========
Current elements in queue: 3
Current elements in queue: 3 1
Current elements in queue: 3 1 5
Current elements in queue: 1 5
Current elements in queue: 5

========== Example 2 ==========
Current elements in queue: 3 1 5
The head element in queue: 3
The tail element in queue: 5
Current elements in queue:
The head element in queue: -1
The tail element in queue: -1

========== Example 3 ==========
Current elements in queue: 3 1 5
Current capacity of queue: 3
Current size of queue: 3
Current elements in queue: 3 1 5 4
Current capacity of queue: 6
Current size of queue: 4
```

- ## Second part: Stack

You must use the **stack** from the standard library to implement, **or you'll get 0 score**.
Do not change the names of the functions and the class.
Please complete the functions in public.
Please implement the queue **BY YOURSELF**.
You can only use `**std::stack**`. Do not use other built-in data structures (like std::queue, std::vector)

Assume all the input data are **positive integers**.
The operations are listed below:

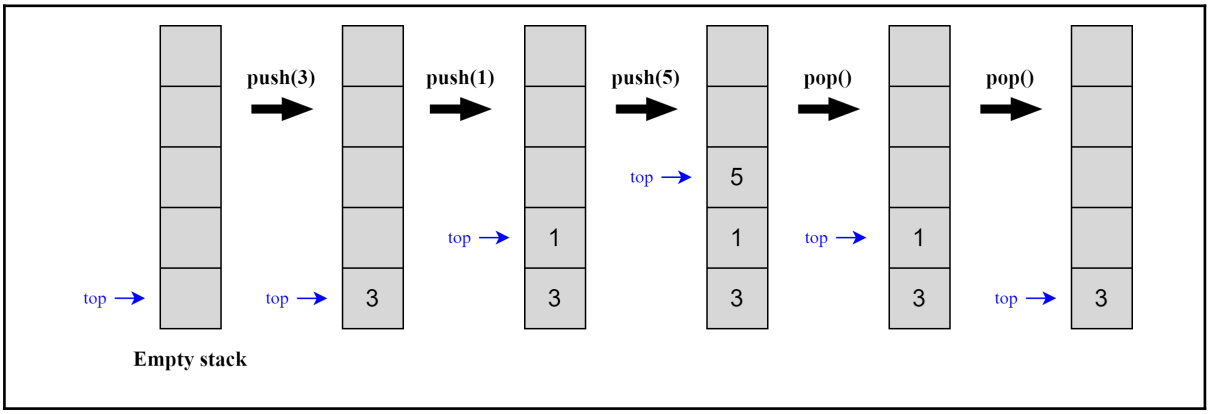
Operations	**Description**
A. (5%) push(x)	Add an element **x** at the end of the queue.
B. (5%) pop()	Remove an element at the head of the queue. If there is no element in the queue, do nothing.
C. (5%) top()	Return the value of the element at the head of the queue. If there is no element in the queue, return -1.
D. (5%) print()	Print the values of the elements in the queue from front to back.

You can refer to the built-in functions of `std::stack` below to implement the second part.

Operations	**Description**
A. stack<int> s	Variable declaration.

B. push(x)	Add an element **x** at the top of the stack.
C. pop()	Remove an element at the top of the stack.
D. top()	Return the value of the element at the top of the stack.
E. empty()	Return true if the stack is empty. Otherwise return false.

Example:



Input:

Use main_Q2B.cpp to test your code.
Don't modify the content of main_Q2B.cpp.

Output:

This is the result of running the main_Q2B.cpp.
You should follow the output format.

```
========== Example 1 ==========
Current elements in queue: 3
Current elements in queue: 3 1
Current elements in queue: 3 1 5
Current elements in queue: 1 5
Current elements in queue: 5

========== Example 2 ==========
Current elements in queue: 3 1 5
The head element in queue: 3
Current elements in queue:
The head element in queue: -1
```