Creative Software Programming Assignment#4 (week-4)

Every assignment will be announced on **Thursday** and should be sumitted by next **Tuesday**.

In this week **Handed out will be Sep 24, 2020, Due Sep 29, 2020**

Structure

- week-4
  - problem1.cc
  - problem2.cc (Optional)

1. Dynamic array (85%)

There is `std::vector` in STL for dynamic array. Using a `std::vector` you can insert or delete elements without regard to size, and random access like you did with arrays.

To create a dynamic array like this `std::vector`, you need to allocate memory and reallocate if the memory size becomes insufficient due to the addition of elements.

In this assignment, implement `len`, `push_back` and `pop_back` (*see also comment of each function*). You can use(or not use) `initialize` and `release` function for allocate/release memory.

More like a Python List

- array is pointer of array.
- cap is allocated size of array.
- In scoring, only call push_back and pop_back
- initialize/release is not called in main

The main function will be removed in scoring

```
#include <iostream>

int* array;
size_t cap = 0;

void initialize(int cap_) {
    array = new int[cap_];
    cap = cap_;
}

void release() {
    delete[] array;
}

void push_back(int element) {
    // TODO:
    // insert element to back of array
    // if array is [1,2,3], and push_back(4) called,
    // then array should be [1,2,3,4]
```

```
}

int pop_back() {
    // TODO:
    // return last element of array and remove it from array
    // if array is [1,2,3,4] and pop_back() called,
    // then array should be [1,2,3] and pop_back() return 4.
}

int len() {
    // TODO:
    // return size of array
}

int main() {
    return 0;
}
```

2. Dynamic array structuer (15%)

With problem-1 we can use template and structs to create dynamic arrays that work in more general case.

implement below function with TODO tag

The main function will be removed in scoring

```
#include <iostream>

template <typename T>
struct dynamic_array {
    T* pointer = nullptr;
    size_t cap = 0;

public:
    dynamic_array(size_t cap)
    : cap(cap) {
        this->pointer = new T[cap];
    }

    void push_front(const T& element) {
        // TODO: push_front
        // if array is [2,3] and pop_front(1) called
        // then array should be [1,2,3]
    }

    void push_back(const T& element) {
        // TODO: push_back
        // if array is [1,2] and pop_front(3) called
        // then array should be [1,2,3]
    }

    T pop_front() {
        // TODO: pop front
        // if array is [1,2,3], and pop_front called
        // then array should be [2,3] and return 1
    }
```

```
    T pop_back() {
        // TODO: pop back
        // if array is [1,2,3], and pop_front called
        // then array should be [1,2] and return 3
    }

    ~dynamic_array() {
        delete[] this->pointer;
    }
};

int main() {
    return 0;
}
```

```
```