Dynamic Arrays

Arrays allocated on the heap with the new operator. Can also use the std::nothrow version of new

Array dynamic allocation

```
size_t size{10};

//Different ways you can declare an array
//dynamically and how they are initialized

double *p_salaries { new double[size]}; // salaries array will
                                        //contain garbage  values
int *p_students { new(std::nothrow) int[size]{} }; // All values initialized to 0

double *p_scores { new(std::nothrow) double[size]{1,2,3,4,5}}; // Allocating memory space
                                                               // for an array  of size double
                                                               //vars. First 5 will be initialized
                                                               //with 1,2,3,4,5, and the
                                                               //rest will be 0's.
```

```
//nullptr check and use the allocated array
if(p_scores){
    //Print out elements. Can use regular array access notation, or pointer arithmetic
    for( size_t i{}; i < size ; ++i){
       std::cout << "value : " << p_scores[i] << " : " << *(p_scores + i) << std::endl;
    }
}
```

Releasing memory
[Array version]

```
delete[] p_scores;
p_scores = nullptr;

delete[] p_students;
p_students = nullptr;

delete[] p_salaries;
p_salaries = nullptr;
```

Pointers and arrays are different

```
//Pointers initialized with dynamic arrays are different from arrays :
//std::size doesn't work on them, and they don't support range based for loops

double *temperatures = new double[size] {10.0,20.0,30.0,40.0,50.0,60.0,70.0,80.0,90.0,100.0};

//std::cout << "std::size(temperatures) : " << std::size(temperatures) << std::endl;//Error

//Error : temperatures doesn't have array properties that are needed for
// the range based for loop to work.
for (double temp : temperatures){
    std::cout << "temperature : " << temp << std::endl;
}

//We say that the dynamically allocated array has decayed into a pointer
```