POO LAB 1

Write the class Phrase so that the following code

```
int main() {
	Phrase p = "I   have   POO   examination   today in C++ ";
	std::cout << "Total words: " << (int)p << "\n";
	for (int i = 0; i < (int)p; i++)
	{
		std::cout << "Word[" << i << "] = " << p[i] << "\n";
	}
	std::cout << "Letter 'a' is present in word: '" << p[3] << "' for " << p.CountLetter(3, 'a') << " times\n";
	std::cout << "Letter 'a' is present in the phrase for " << p.CountLetter('a') << " times\n";
	std::cout << "Longest word is: " << p.GetLongestWord() << "\n";
	std::cout << "Total vowels: " << p.CountVowels() << "\n";
	Phrase p2 = "Hello";
	std::cout << "Words in p2: " << (int)p2 << "\n";
	Phrase p3 = "  hello    world   ";
	std::cout << "Words in p3: " << (int)p3 << " [" << p3[0]<<" and "<< p3[1]<<"]\n";
	return 0;
}
```

compiles and upon execution prints the following to the screen:

```
Total words: 7
Word[0] = I
Word[1] = have
Word[2] = POO
Word[3] = examination
Word[4] = today
Word[5] = in
Word[6] = C++
Letter 'a' is present in word: 'examination' for 2 times
Letter 'a' is present in the phrase for 4 times
Longest word is: examination
Total vowels: 14
Words in p2: 1
Words in p3: 2 [hello and world]
```

Observations:

1. You are not allowed to use **STL** at all (for vectors, strings, maps or any **template/object** defined in STL). The only exception is the usage of "**std::cout**" from the main function
2. You are not allowed to use string manipulation functions defined in "string.h" such as **strlen**, **strcpy**, **strdup, strtok**, etc. You can however use **memcpy** if you need.
3. You are not allowed to use **malloc** / **calloc** and / or **free** functions. Use new and delete instead.
4. If you don't respect the previous conditions (e.g. use strlen, or strcpy, etc) → we will compute the correctness of the code, but the final grade will be half of the computed score. For example, if you do everything corect, but you use STL or string based functions at least one time, you will receive 15 points instead of 30.
5. The string that is passed to the Phrase constructor is formed out of words that are separated by one or multiple spaces. A word is formed out of letters, numbers or symbols such as '+', '-', etc.

Grading:

G1	Writing the Phrase constructor that splits a phrase into words and stores the words in an internal array.	7p **AP**
G2	Phrase destructor	1p **AP**
G3	Organize your project in 3 files: **main.cpp**, **Phrase.h** and **Phrase.cpp**	1p **WP**
G4	Organize your class Phrase to include private and public members, the definition of a constructor and destructor, and at least one method.	2p **WP**
G5	Operator to cast to int that returns the total number of words	1p **AP**
G6	Operator[] that returns the word with a specific index	1p **AP**
G7	Method: CountLetter(int, char) → the number of times a specific character is found in a word from the phrase	3p
G8	Method: CountLetter(int) → the number of times a specific character is found in the phrase	3p
G9	Method: CountVowels() → the number of vowels (including capital letters) are found in the phrase	3p
G10	GetLongestWord() → returns the longest word in the phrase	5p
G11	Program compiles and upon execution produces the expected results	3p

Criteria explanation:

AP ⇒ the capability to correctly apply OOP principles (inheritance, polymorphism, etc)

WP⇒ the capability to write C++ programs based on specifications

The evaluation will consider all the relevant points for a criteria.