Computer Lab #3.

Implementation of a Container Data Structure in C and C++ Programming Languages. Constants in C++.

Task 1. Copy the source code provided below to your C/C++ development environment and try to execute it. Did you get any error messages? Explain them. What pieces of code below are incorrect and why?

```
void main()
{
	// 1.
	const int i;

	// 2.
	const int i=90;
	i++;

	// 3.
	const int i=90;
	int *p=&i;
	(*p)++;

	// 4.
	const int v[]={1,2,3};
	v[1]++;

	// 5.
	const int i=255;
	int v[i];

	// 6.
	char s[]="Hello";
	const char *pc=s;
	pc[0]='h';
	pc++;

	// 7.
	char s[]="Hello";
	char* const cp=s;
	cp[0]='h';
	cp++;

	// 8.
	char s[]="Hello";
	const char* const cpc=s;
	cpc[0]='h';
	cpc++;

	//9.
	int j = 0;
	int const &i = j;
	i = 1;
	const int &i = j;
	i = 1;
```

```
}
```

Task 2. Download the ***CL3_files.zip*** file from the Course Webpage, open the archive and copy the ***stack.c*** file to your C/C++ development environment. Complete the given source code file to realize a stack data structure that can work with *char* type elements. Follow the comments and the instructions in the code.

General conditions and constraints.

1. If the stack is empty, then *pData* should be NULL and *elements* should be 0.
2. *pData* points to an array of elements of variable length; dynamic memory allocation should be used.
3. The *elements* variable denotes the actual number of elements (data) in the stack.
4. Before using the stack the following initialization function must be called:
   void stack_init(struct stack s).*
5. After having used the stack the following function must be called:
   void stack_cleanUp(struct stack s).*
6. If an error occurs then the return value of any local function must not be 0.

Task 3. Consider simpler parameter passing: replace pointers with C++ references wherever possible. Remember to use the C++ compiler. Check the correctness of your replacements at execution time in the debugger mode.