Name: Kormoua Khongmeng
Neptun Code: I3MLPQ

Basic of Programming 2

Laboratory report 3

Task1: 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() {
```

Ans: we cannot use **void main** in C++ because the standard does not allow it. **main()** function should return integer. So it must be "**int main()**".

```
// 1.
const int i;
```

Ans: i is declared to be constant integer value but we have not assigned any value for it. We must assign some value for constant **i** here.

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

Ans: an error here, we redefined **i** here which has been declared earlier in above problem.

```
i++;
```

Ans: i here is defined to be a constant value which will not be able to be modified or change its value after initialization. But here we tried to increment it, so it generates an error.

```
// 3.
const int i = 90;
```

Ans: an error here, we redefined **i** here which has been defined earlier in above problem.

```
int *p = &i;
```

Ans: found an error here, we cannot set a non-constant pointer to a constant variable. We must do like this instead: **const int *p = &i;**

```
(*p)++;
```

Ans: found an error here, **i** is a constant variable which will not be able to change its value. A pointer which can point to it will also be a constant pointer which behave like a read-only variable. So we cannot increment a value of **i** like this.

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

Ans: array **v** contains a constant int which cannot be modified, so when we try to modify its element like this case, it generates error.

// 5.
const int i = 255;
Ans: an error here, we redefined **i** again which has been defined earlier above.

int v[i];
Ans: an error here, we redefined an array **v** again which have been defined earlier at the above problem.

// 6.
char s[] = "Hello";
const char *pc = s;
pc[0] = 'h';
Ans: an error here, **pc** here behaves like a read-only variable which is not assignable.

pc++;

// 7.
char s[] = "Hello";
Ans: an error here, we redefined a string **s** again which have been defined earlier at the above problem.

char* const cp = s;
cp[0] = 'h';
cp ++;
Ans: an error here, **cp** is a constant pointer which cannot be assigned. In other word, **cp** can point to only one place and can't be pointed to other places again.

// 8.
char s[] = "Hello";
Ans: an error here, we redefined a string **s** again which have been defined earlier at the above problem.

const char* const cpc = s;
cpc[0] = 'h';
Ans: an error here, **cpc** in this case behaves like both a read-only variable and cannot point to anywhere else too after initialization. Here we tried to modify a value that **cpc** points to which is not possible.

cpc++;
Ans: an error here, **cpc** in this case behaves like both a read-only variable and cannot point to anywhere else too after initialization. Here we tried to assign some value to the pointer, in other word we tried to point this pointer to some other space which is not possible for **cpc** here.

// 9.
int j = 0;
int const &i = j;
Ans: an error here, we redefined **i** again which has been defined earlier in the above problem.

i = 1;
Ans: i here refers to another int, which cannot be modified through this reference. However that int can be modified elsewhere.

const int &i = j;
Ans: an error here, we redefined **i** again which has been defined earlier.

i = 1;
Ans: i here refers to another int, which cannot be modified through this reference. However that int can be modified elsewhere.
}

Task2: 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.

Ans: The solution will be in the same folder name as “solution2.cpp”

Task3: 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.

Ans: The solution will be in the same folder name as “solution3.cpp”