```
Script started on 2024-12-05 11:53:27-06:00 [TERM="xterm-256color" TTY="/dev/pts/6'
e_prevost@ares:~/Lab_2_p3$ pwd
/home/students/e_prevost/Lab_2_p3
e_prevost@ares:~/Lab_2_p3$ cat digit_manipulator.info
***************

Robert Prevost

CSC 122 W01



longobjops lab



extends the basic functionality associated with a long integer



Base Level: Level 2.5

Total Level: Level 2.5

***************e_prevost@ares:~/Lab_2_p3$ show-code digit_manipulator.cpp


digit_manipulator.cpp:
```

```cpp
#include <iostream>
#include <cmath>
using namespace std;

class DigitExtractor {
    private:
        long value;

        int getDigitCount() const {
            if (value == 0) return 1;
            return floor(log10(value)) + 1;
        }

    public:
        DigitExtractor(long val = 0) : value(val) {}

        operator long() const {
            return value;
        }

        long operator[](long position) const {
            int digits = getDigitCount();
            int pos = floor(log10(position));
            if (pos >= digits) return -1;

            return (value / (long)pow(10, pos)) % 10;
        }

        long operator()(long start, long end) const {
            int startPos = floor(log10(start));
            int endPos = floor(log10(end));
            int digits = getDigitCount();

            if (startPos >= digits) return -1;

            if (startPos < endPos) {
                int temp = startPos;
                startPos = endPos;
                endPos = temp;
            }

            long divisor = pow(10, endPos);
            long result = value % (long)pow(10, startPos + 1);
            result /= divisor;

            return result;
        }

        friend ostream& operator<<(ostream& out, const DigitExtractor&
        obj) {
            out << obj.value;
            return out;
        }

        friend istream& operator>>(istream& in, DigitExtractor& obj) {
            in >> obj.value;
            return in;
        }
};

int main() {
    DigitExtractor obj(1234);

    cout << obj[10] << endl;
    cout << obj[1000] << endl;
    cout << obj[10000] << endl;

    cout << obj(10,1000) << endl;
    cout << obj(1000,10) << endl;
    cout << obj(1,10000) << endl;
    cout << obj(1,1) << endl;
    cout << obj(1,100) << endl;
    cout << obj(10,100) << endl;
    cout << obj(10000,100000) << endl;
    return 0;
}
```

```
e_prevost@ares:~/Lab_2_p3$ CPP digit_manipulator
digit_manipulator.cpp***
```

```
digit_manipulator.cpp: In member function ‘int
DigitExtractor::getDigitCount() const’:
digit_manipulator.cpp:11:40: warning:
conversion from ‘double’
to ‘int’ may change value
[-Wfloat-conversion]
   11 |             return floor(log10(value)) + 1;
      |                    ~~~~~~~~~~~~~~~~~~~~^~~
digit_manipulator.cpp: In member function ‘long
int DigitExtractor::operator[](long int) const’:
digit_manipulator.cpp:23:28: warning:
conversion from ‘double’
to ‘int’ may change value
[-Wfloat-conversion]
   23 |             int pos = floor(log10(position));
      |                       ~~~~~^~~~~~~~~~~~~~~~
digit_manipulator.cpp:26:46: warning:
use of old-style cast to ‘long int’
[-Wold-style-cast]
   26 |             return (value / (long)pow(10,
   pos)) % 10;
      |                                              ^
      |                                 ------------------
      |                                 static_cast<long>
      (pow(10, pos))
digit_manipulator.cpp: In member function ‘long
int DigitExtractor::operator()(long int, long int) const’:
digit_manipulator.cpp:30:33: warning:
conversion from ‘double’
to ‘int’ may change value
[-Wfloat-conversion]
   30 |             int startPos =
   floor(log10(start));
      |                           ~~~~~^~~~~~~~~~~~~~
digit_manipulator.cpp:31:31: warning:
conversion from ‘double’
to ‘int’ may change value
[-Wfloat-conversion]
   31 |             int endPos = floor(log10(end));
      |                          ~~~~~^~~~~~~~~~~~
digit_manipulator.cpp:42:31: warning:
conversion from ‘__gnu_cxx::__promote_2<int, int,
double, double>::__type’ {aka ‘double’}
to ‘long int’ may change value
[-Wfloat-conversion]
   42 |             long divisor = pow(10, endPos);
      |                            ~~~^~~~~~~~~~~~
digit_manipulator.cpp:43:61: warning:
use of old-style cast to ‘long int’
[-Wold-style-cast]
   43 |             long result = value % (long)pow(10, startPos +
   1);
      |
      ^
      |                                      ------
      |                                      static_cast<long>
      (        )


e_prevost@ares:~/Lab_2_p3$ ./digit_manipulator.out
3
1
-1
123
123
1234
4
234
23
-1
e_prevost@ares:~/Lab_2_p3$ cat digit_manipulator.tpq
Which operators are friends and which are members? Do any have to be

members?



stream operators are friends because they arent naturally members of the

class. The subscript and function call operators must be members.



Which operators should be const? What other methods might well be const?



the subscript operator, function call operator and  typecast operator

should all be const, object is not modified in these functions.



Does this class serve any useful purpose? Why/Why not?



Yes it is useful for robust data manipulation where you need to extract

digits from a long number.



What use is a typecast operator? When would it be called? Why would you

want your objects to be cast back to a simpler type?
```

```
typecast operator allows the class to work along with toher long digits.

and if we have a parameter in another function that requires a long we can

use the typecast operator to insert a long into that parameter. e_prevost@ares:~/La
exit

Script done on 2024-12-05 11:54:05-06:00 [COMMAND_EXIT_CODE="0"]
```