```cpp
// Programming in C/C++
// Week 2: Assignment 11
// Tjalling Otter & Emiel Krol

#include <iostream>
using namespace std;

int main(int argc, char *argv[])
{
  size_t number = stoul(argv[1]);                // Initialise and enter the integer

  for(size_t index = 1; index != 11; index++)    // For the numbers 1-10
    cout << index << " * " << number << " = "    // Display those multiplied by
         << (index * number) << '\n';            // the integer
}
```

FB

5

pp

SF.

++index

?11

```cpp
// Programming in C/C++
// Week 2: Assignment 12
// Tjalling Otter &  Emiel Krol

#include <iostream>
#include <string>

using namespace std;

int main()
{
  string asciiSet;                                  // Initialise the ascii string

  for (size_t index = 0; index < 255; index++)      // Loop through all ascii characters
  {
    if (isalpha(index))                             // If they are alphabetical ...
    {
      asciiSet += (char) index;                     // ... add them to the string
    }
  }
  cout << asciiSet << '\n';                         // Print the alphabetical ascii set
}

```

```cpp
// Programming in C/C++
// Week 2: Assignment 13
// Tjalling Otter & Emiel Krol

#include <iostream>
using namespace std;

int main(int argc, char *argv[])
{
  size_t numberOfLines = 0;                    // Initialise integer
  string extractedLine;                        // Initialise string

  if (argc == 2 && string(argv[1]) == "ok")    // If optional ok is given
  {
    while (true)                               // Infinite loop
    {
      if (!getline(cin, extractedLine))        // Until no more lines
        break;
      numberOfLines++;                         // Increment integer
    }
  }
  else
  {
    while (!cin.eof())                         // If there is a line (also enter it)
    {
      getline(cin, extractedLine);             // Get line (again)
      numberOfLines++;                         // Increment int
    }
  }
  cout << numberOfLines << '\n';               // Output the int
}

// Run with ./exercise13 [ok] < ./fileForExercise13.txt, which has five lines
```

b

pp

wat ook werkt is argv[1] = "ok"'s

mind the PP!

#13

```cpp
// Programming in C/C++
// Week 2: Assignment 14
// Tjalling Otter &  Emiel Krol

#include <iostream>
#include <string>


using namespace std;

int main()
 {
  string line;
  string text;
  string letter;
  string lastword;

  cout << "Enter line, type ^C after your last line.\n";
  while(getline(cin,line))                    //input for multiple lines
  {
    if(line=="^C")
    break;

    size_t count = line.length()-1;           //Setting the counter to the
                                              //last index of the string.
    lastword.clear();
    while (!isspace(line[count]))             //Finds the last word of the line
    {
      letter = line[count];

      if(letter.compare(".") != 0)            //Keeping the period at the end of the line
        lastword.insert(0,letter);            //Saving the last word of the line

      count--;
    }

    count = 0;                                //resetting the index counter to 0
    while (!isspace(line[count])&&line.find(" ")!= string::npos)  //deleting the first word
    {                                                             //if more than one
      line.erase(0,1);
    }

    if(line.find(" ")!= string::npos)         //Testing whether line is empty
      line = lastword + line;                 //Adding the last word to the
                                              //start of the line if it is not

    text = text + line + '\n';                //Adding the line to the rest of the text
  }
  cout << text;                               //Printing output.
}
```

use string members
and ask yourself how you would
solve the problem. Then implement that...

```cpp
// Programming in C/C++
// Week 2: Assignment 15
// Tjalling Otter &  Emiel Krol

#include <iostream>
#include <string>
#include <sstream>
#include <fstream>

using namespace std;
//use with textEx15.txt

int main()
{
  unsigned int totallines =0;
  unsigned int linenumber =0;
  unsigned int emptylines =0;
  unsigned int lastlinewithtext =0;

  bool islastline = false;                  //If a line without text is read this value
  becomes 1,
                                            //if a line with text is read afterwards it
                                            becomes 0 again.
  bool isfirstline = false;                 //simular for first line with text.

  ifstream text;
  text.open("textEx15.txt");

  string line;
  while (getline(text,line))                //while to find first and last line with text
  {
    totallines++;
    if(line.empty()&(islastline==false))    //finding last line with text
    {
      lastlinewithtext = totallines;
      islastline=true;
    }
    if(!line.empty())
    {
      islastline=false;
    }
    if(line.empty()&(isfirstline==false))   //finding first line with text
    {
      emptylines++;
    }
    if(!line.empty()&(isfirstline==false))
    {
      isfirstline=true;
    }
  }

  text.close();
  text.open("textEx15.txt");                //closing and opening the file so we can now
  use the filter.

  while (getline(text,line))                //while loop to print text where the filter
  values are determined
  {                                         //in the while loop above.
    linenumber++;
    if(emptylines < linenumber)             //removes whitespace above initial text
    {
      if(lastlinewithtext > linenumber)     //lastlinewithtext > currentline
      {
        cout << line <<'\n';
      }
    }
  }

}
```

```
Programming in C/C++
Week 2: Assignment 17
Tjalling Otter & Emiel Krol

*******************************begin code*******************************
// Programming in C/C++
// Week 2: Assignment 17
// Tjalling Otter & Emiel Krol

#include <iostream>
#include <string>
#include <cmath>

using namespace std;

int main(int argc, char const *argv[]) {

  string input = argv[1];                 // the first argument is the binary number
                                          //supplied

  size_t value = stoi(input,nullptr,2);   //Turning the string into an integer so
                                          //we can do calculations.
                                          //Where the 2 indicates that the string
                                          //is a binary number.
  size_t newValue;

  size_t digits = input.length();         //Number of digits in the supplied binary number

  size_t caseint;                         //For checking if argv[2] is rol or ror

  string stringrorofrol = argv[2];        //Saving argv[2] in a string to test
                                          //whether it is rol or ror

  if(stringrorofrol=="ror")               //Checking whether argv[2] is ror
    caseint=0;

  if(stringrorofrol=="rol")               //Checking whether argv[2] is rol
    caseint=1;


  switch (caseint) {
    case 0:                                       //if ror
      if(((value>>1)<<1)==value)                  //if even
      {
        newValue = (value>>1);                    //example 1000 -> 0100 ror
        break;
      }
      if(((value>>1)<<1)!=value)                  //if uneven
      {
        newValue = (value>>1)+pow(2,digits-1);    //example 1001 -> 1100 ror
        break;
      }
    case 1:                                       //if rol
      if((value-pow(2,digits-1))>= 0)             //if most significant digit 1
      {
        newValue = (value<<1)+1-pow(2,digits);    // example 1000 -> 0001 rol
        break;
      }
      else //if most significant digit 0
      {
        newValue = (value<<1);                    // example 0101 -> 1010 rol
        break;
      }
    break;
    }

  string newBit;                  //new string to store our new bit value
  size_t bitV = newValue;         //variable that we can alter to find the binary number

  for (digits; 0 < digits; digits--) { //Checking if most significant bit should
    if(bitV-pow(2,digits-1)<0)         //be a 1 or a 0. And adding it to the string.
```

```
73        {                                    //then the same for the most significant bit
74          newBit.append("0");                //after that...until there are no more bits.
75        }
76        if(bitV-pow(2,digits-1)>=0)
77        {
78          newBit.append("1");
79          bitV = bitV - pow(2,digits-1);
80        }
81      }
82
83      //Printing the value in binary - decimal and hexadecimal
84      cout << newBit << ' ' << newValue << ' ' << hex << newValue << '\n';
85    }
86
87    ******************************End code******************************
88
89    input:      ./rotate 1011110101011011 ror
90    output:     1101111010101101 57005 dead
91    input:      ./rotate 1101111101110111 rol
92    output:     1011111011101111 48879 beef
93
94    ************************************************************************
95
```

TC: use += and '0', if you need this

```cpp
// Programming in C/C++
// Week 2: Assignment 18
// Tjalling Otter & Emiel Krol

// Needs C++17

#include <iostream>
using namespace std;

enum commandOptions                                 // enum representations of
{                                                   // command options
  store, add, subtract, multiply, divide, ret, invalid
};

commandOptions matchEnum(string const& command) // enums for the commands
{
  if (command == "sto") return store;               // It is not very pretty,
  if (command == "add") return add;                 // but the switch statement
  if (command == "sub") return subtract;            // really wants integers
  if (command == "mul") return multiply;
  if (command == "div") return divide;
  return invalid;
};

int main()
{

  size_t programVariable = 0; // Initialise the program's variable

  cout << "Welcome to fake assembly \n";  // Welcome message

  while(true) // Infinite loop to create interactive prompt
  {
    string command; // Initialise the command string
    size_t parameter; // Initialise the numerical parameter

    cout << "> "; // Input prompt

    cin >> command; // First input is the command (string)
    if(command == "ret")  // Quick check if the user wants to exit
    {
      cout << "Program has ended. The variable ended with the value " << programVariable <<
      '\n';
      break;
    }
    else // If not, the second word is the parameter
    {
      cin >> parameter;
    }

    switch (matchEnum(command))        // Switch statement for the commands
    {
      case store:                      // Store (sto) command
          programVariable = parameter;
        break;
      case add:                         // Add (add) command
        programVariable += parameter;
        break;
      case subtract:                    // Subtract (sub) command
          programVariable -= parameter;
        break;
      case multiply:                    // Multiply (mul) command
        programVariable *= parameter;
        break;
      case divide:                      // Divide (div) command
        if(parameter == 0)              // With check for /0
        {
          cout << "No instruction 'store' \n";
          break;
        }
        else
        {
```

```
72              programVariable /= parameter;
73              break;
74            }
75          case invalid:
76            [[fallthrough]];
77          default:                              // Invalid input
78            cout << "Invalid input. \n";
79            cin.clear();                        // Clearing unread input, otherwise
80            cin.sync();                         // there will be an infinite loop
81            break;
82      }
83    cout << programVariable << '\n';            // Output of variable
84    }
85  }
86
```

```cpp
// Programming in C/C++
// Week 2: Assignment 19
// Tjalling Otter &  Emiel Krol

#include <iostream>
#include <string>

using namespace std;

int main(int argc, char *argv[])
{
  if (argc != 3)
  {
    cout << "This program expects two command-line arguments. Exiting. \n";
    return(0);
  }

  size_t base = stoul(argv[1];            // First argument, radix
  size_t inputValue = stoul(argv[2]);     // Second argument, number to convert
  size_t intermediaryValue = inputValue;  // Initialise intermediary value
  string outputValue;                     // Initialise output string

  if (inputValue == 0)  // Quick exit for when 0 will stay 0 in any base
  {
    outputValue = "0";
  }

  while (intermediaryValue != 0)  // Continuous loop while initial put is decremented
  {
    size_t prependValue;  // Initialising the digit to prepend to the string

    prependValue = (intermediaryValue % base);              // Finding out the remainder
                                                            // of a division
                                                            // to prepend to the string
    if (prependValue > 9)                                   // If remainder > 9, ...
    {
      outputValue.insert(0, 1, (char) (prependValue += 87)); // ... it must be an
                                                            // alphabetical character
    }
    else                                                    // If not, ...
    {
      outputValue.insert(0, 1, (char) (prependValue += 48)); // .. it is just a number
    }
    intermediaryValue /= base;  // Decrementing the value to move to the next digit
  }
  cout << inputValue << ", displayed using raxix " << base << " is: " //Output
       << outputValue << '\n';
}
```