```
Script started on Thu 23 Feb 2017 09:03:11 PM CST
\033]0;b_pepa@mars:~/CSC122/assigned_problems\007[b_pepa@mars assigned_problems]$ pwd
/home/students/b_pepa/CSC122/assigned_problems
\033]0;b_pepa@mars:~/CSC122/assigned_problems\007[b_pepa@mars assigned_problems]$ cat
a\033[Kssigned.info
Brandon Pepa
CSC122-001
You want me to do what problems?
Lab

Levels:
(level 4)
(level 2)
        encapsulate into a class

**(level 6)**

Description:
        This function takes the input of a name and a list of problems
and turns it into a sortes list of problems that is easy to read exactly
what problems someone is required to do for a class.
\033]0;b_pepa@mars:~/CSC122/assigned_problems\007[b_pepa@mars assigned_problems]$ cat
assigned.cpp
#include <iostream> //cin&cout
#include <vector> //store problem numbers
#include <cmath> //log function to find # of chars in a number
#include <string> // Multiple use of strings
using namespace std;

//deletes char a[index]
void delChar(string &a, unsigned index);

//Returns the first number(can be multiple digits) in a string
long get1stNum(string &a);

//Returns true if it finds the value in the list (linear find)
bool findNum(vector<long> &a, long search);

class problem_set
{
      vector<long> problem_nums;
      string problem_name;
      string user_input;
      void setName(string & a);
      void setProblems(string & a);
      void insertSort(long input);
   public:
      problem_set() : problem_nums(), problem_name(), user_input() {}
      void read(void);
      void print_final(void);
      void print_input(void);

};


int main(void)
{
   problem_set problem;
   problem.read();
   problem.print_input();
   problem.print_final();
return 0;
}


void problem_set :: read(void)
{
   string temp; //Temporary string to modify so original input remains
                //Protected (can be output if need be)

   cout << "Please Enter the problem set numbers you are" << endl
        << "required to complete, then press enter." << endl;
   getline(cin, user_input);
   temp = user_input;
   setName(temp);
   setProblems(temp);
return;
}

void problem_set :: setName(string & a)
{
        //removes any preceding spaces or quotes
   while(isspace(a[0]) || a[0] == '\'' || a[0] == '\"')
   {
      a = a.substr(1);
   }

        //creates the name until it finds an end quote
        //or until it finds a comma (but the comma will be
        //included temporarily so we can deal with it)
   unsigned i = 0;
   while(a[i] != '\''  && a[i] != '\"' &&
         a[i-1] != ',' && a[i-1] != '-'&&
         i+1 != a.length())
   {
      problem_name += a[i];
      i++;
   }
   if(a[i] == '\'' || a[i] == '\"')
   {
      delChar(a,i);
   }

        //if we didn't detect an end quote
   if(problem_name[problem_name.length()-1] == ',' ||
      problem_name[problem_name.length()-1] == '-')
   {
        //loops till the last character isn't a space or
        //a number
      do
      {
         problem_name = problem_name.substr(0,problem_name.length()-1);
      }
      while(isdigit(problem_name[problem_name.length()-1]) ||
            isspace(problem_name[problem_name.length()-1]) );
   }

        //removes the problem name from the string we read
   a = a.substr(problem_name.length());
return;
}

void problem_set :: setProblems(string & a)
{
   long temp,
        temp2;

        //If there's a quote left at the beginning, remove it
```

```
   if(a[0] == '\'' || a[0] == '\"')
   {
      a = a.substr(1);
   }

   while(!a.empty())
   {
      temp = get1stNum(a);

      if(isspace(a[0]))
      {
         a = a.substr(1);
      }

        //how to deal with hyphenated problems
      if(a[0] == '-')
      {
         a = a.substr(1);
         temp2 = get1stNum(a);
         for(int i = temp; i <= temp2; i++)
         {
            if(!findNum(problem_nums,i))
            {
               insertSort(i);
            }
         }
      }
        //if there was not a hypehn do this
      else if(!findNum(problem_nums,temp))
      {
         insertSort(temp);
      }

        //shorten the string till it finds the next problems
        //or until there's nothing left
      while(a[0] != ',' && a.length() != 0)
      {
         a = a.substr(1);
      }
      if(a[0] == ',')
      {
         a = a.substr(1);
      }
   }

return;
}

void problem_set :: print_input(void)
{
   cout << "You entered " << user_input << endl;
}

void problem_set :: print_final(void)
{
   //represents #of characters in output stream
   //("do problem " has 11 characters)
   long chars = 11;

   cout << "Do problem";
   if(problem_nums.size() > 1)
   {
      cout << "s";
      chars++;
   }
   cout << " ";

   if(problem_nums.size() == 1)
   {
      cout << problem_nums[0];
      chars += static_cast<long>(log10(problem_nums[0]) + 1);
   }
   else
   {
      for(unsigned i = 0; i + 1 < problem_nums.size(); i++)
      {
         cout << problem_nums[i] << ", ";
         chars += static_cast<long>(log10(problem_nums[i]) + 1);
         chars += 2;

         // if there are more than 70 characters that have been output
         // start a new line
         if(chars >= 70)
         {
            chars = 0;
            cout << endl;
         }
      }
      if(!problem_nums.empty())
      {
         cout << "and " << problem_nums[problem_nums.size()-1];
      }
   }

   cout << " of " << problem_name << "." << endl;
return;
}

void problem_set :: insertSort(long input)
{
       //If there's nothing in the list yet, pushback
   if(problem_nums.empty())
   {
      problem_nums.push_back(input);
   }
   else
   {
      //Find where the number needs to be inserted
      unsigned position = 0;
      while(position + 1 <= problem_nums.size() &&
            input > problem_nums[position])
      {
         position++;
      }


      //Insert the number into position
      problem_nums.resize(problem_nums.size()+1);

      for(unsigned i = problem_nums.size()-1; i > position; i--)
      {
         problem_nums[i] = problem_nums[i-1];
      }
      problem_nums[position] = input;
   }

}
```

```
long get1stNum(string &a)
{
   long temp = atol(a.c_str());
   while(a[0] != '-' && a[0] != ',' &&
         a.length() > 1)
   {
      a = a.substr(1);
   }
   if(a.length() == 1)
   {
      a = "";
   }
return temp;
}

void delChar(string &a, unsigned index)
{
   a = a.substr(0, (a.length()-index)) +
       a.substr(index+1);
}

bool findNum(vector <long> &a, long search)
{
   if(a.size() < 1)
   {
      return false;
   }
   for(unsigned i = 0; i + 1 < a.size(); i++)
   {
      if(a[i] == search)
      {
         return true;
      }
   }
   return false;
}
\033]0;b_pepa@mars:~/CSC122/assigned_problems\007[b_pepa@mars assigned_problems]$ CPP
assigned
assigned.cpp***
assigned.cpp:18:   instantiated from here
assigned.cpp:18:   instantiated from here
assigned.cpp:18:   instantiated from here
assigned.cpp:166:   instantiated from here
assigned.cpp:175:   instantiated from here


\033]0;b_pepa@mars:~/CSC122/assigned_problems\007[b_pepa@mars assigned_problems]$ ./as
signed.out
Please Enter the problem set numbers you are
required to complete, then press enter.
"Lesson" 1-4, 3,15, 20
You entered "Lesson" 1-4, 3,15, 20
Do problems 1, 2, 3, 4, 4, 15, and 20 of Lesson.
\033]0;b_pepa@mars:~/CSC122/assigned_problems\007[b_pepa@mars assigned_problems]$ ./as
signed.out
Please Enter the problem set numbers you are
required to complete, then press enter.
L 1-100;
You entered L 1-100
Do problems 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,
36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53,
54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71,
72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89,
90, 91, 92, 93, 94, 95, 96, 97, 98, 99, and 100 of L.
\033]0;b_pepa@mars:~/CSC122/assigned_problems\007[b_pepa@mars assigned_problems]$ ./as
signed.out
Please Enter the problem set numbers you are
required to complete, then press enter.
'Lesson 15' 1-5, 13, 13,13, 29 , 16
You entered 'Lesson 15' 1-5, 13, 13,13, 29 , 16
Do problems 1, 2, 3, 4, 5, 13, 13, 16, and 29 of Lesson 15.
\033]0;b_pepa@mars:~/CSC122/assigned_problems\007[b_pepa@mars assigned_problems]$ cag
\033[Kt pr\033[K\033[K033[Kassigned.tpq
Thought Provoking Questions

Q1.In what data type(s) can you store something like the problem set name that
   might be anything from one character to a whole bunch of text?

A: strings.

Q2.How can you detect that the problem set name is a quoted string? How can you
   read in the whole thing to a single variable? (Don't worry that the user has
   missed a close quote.)

A: You can check the entire in put for a closed quote (either one) and if it
   doesn't find one you can look for the first comma, look back until there is no
   numbers and have that be the end of the name.

Q3.If the user's problem set name is a quoted string, how can you remove the
   quotes from it? (Just like c++ won't leave your quotes on a string when it
   prints it, you shouldn't leave the user's quotes on the string. They are just
   there to seperate the problem set name from the problem list... and group it
   together as it may contain a space or tab...)

A: If the name is stored as a string with the quotes included, the quotes can be
   simply removed in the next step.

Q4.When placing items (say problem numbers?) into a list, how can you keep those
   items sorted from beginning? (I.E. from their very insertion into the list..?)

A: As the numbers are being placed into the list they can be insertion sorted very
easily

Q5.How can you avoid placing a duplicate item into a list? (Note: i'm not asking
   how you can remove duplicates from a list, but rather how can you avoid having
   any duplicates in the list in the first place!)

A: As the numbers are insertion sorted, if the comparison is equal, the number will
be removed instead of added to the list.

Q6.If the problem list is long (like in math or physics), how can you wrap the long
   output line to multiple lines without too much difficulty? (Hint: the output
   line can be around 70-75 chars long, we saw in 121 how to tell how many digits
   long an integer is, we know we are outputting a comma and a space between the
   problem numbers, and we know how many chars we are adding before the list and
   after the list. Just don't forget the 'and' before the last element. So, knowing
all that, especially that you need to drop to a new line after every 70-ish
   chars you've output, how do you tell if total chars printed is 70*n?)

A: have a count variable, and add however many characters each item adds, and when
   the count variable is between 70 and 75, insert a new line character.
\033]0;b_pepa@mars:~/CSC122/assigned_problems\007[b_pepa@mars assigned_problems]$ exit
exit

Script done on Thu 23 Feb 2017 09:06:04 PM CST
```