Part I: Practice and Theory

The following problems are for practise only and will **not be collected**.

Review problems: R7.1-R7.16.

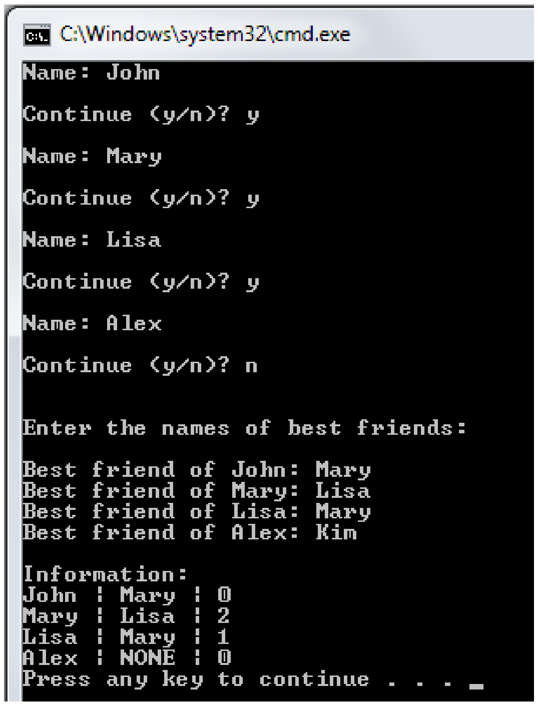
Practice Problems: P7.3, P7.5, P7.9, P7.10, P7.11, P7.12, P7.14.

Part II: Programming.

The following problems will be **collected** and three of them graded.

(1) **Problem P7.1**.

- You can assume that all names in the list of people that the user provides are different.
- Also you can assume that a best friend of a `Person` is not necessarily on the list. If it is the case, the pointer to a best friend must not be be initialized (see the sample of input-output).
- Submit the solution as `hmw_1_1.cpp`
- Sample input-output:



```
Name: John

Continue (y/n)? y

Name: Mary

Continue (y/n)? y

Name: Lisa

Continue (y/n)? y

Name: Alex

Continue (y/n)? n


Enter the names of best friends:

Best friend of John: Mary
Best friend of Mary: Lisa
Best friend of Lisa: Mary
Best friend of Alex: Kim

Information:
John | Mary | 0
Mary | Lisa | 2
Lisa | Mary | 1
Alex | NONE | 0
Press any key to continue . . .
```

(2) Based on **Problem P7.6**

- Write the function

```
double* maximum(double* a, int a_size)
```

described in Problem P7.6. **Do not use `vectors`.**

- Write a program that reads in a list of numbers contiguously and saves them in a dynamically allocated array. **Hint:** Using `getline`, save the list provided by the user in a `string s`. Use the string `s` to count the number of the elements in the list and save this number in `a_size`.

Then dynamically allocate an array of size `a_size`, using the operator `new[]`. After that convert the numbers in the string `s` into floating-point numbers and save them in the array. For conversion one can use the function `std::stod`. For example, `std::stod("1.25")` converts the string `"1.25"` into a floating-point number 1.25.

• Let `double* a` be the pointer associated with the array. Apply the function `maximum` to the array located at `a`. Then display both the maximal element and its index. **Hint**: Use pointer arithmetic. Suppose `p` is the pointer that points to the maximal element, then, in view of the fact that `a` points to the beginning of the array, the difference `p-a` will contain the number of elements between the two pointers, which gives the index.

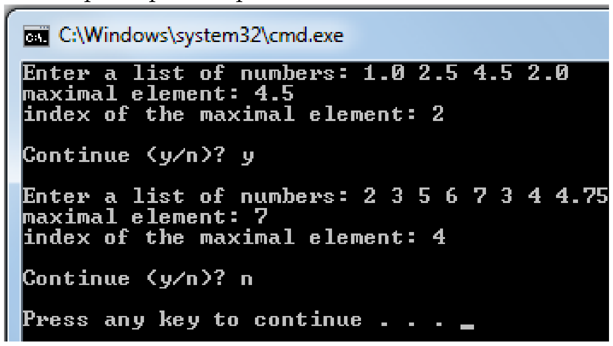
• Implement a loop in which the above actions are repeated until the user requests to quit.

• Assume that the the user's input is always valid.

• Assume that the character after the last number in the input list is the newline character and that there is exactly one space between the numbers in the list.

• Submit the solution in the file named `hmw_1_2.cpp`

• Sample input-output:



```
Enter a list of numbers: 1.0 2.5 4.5 2.0
maximal element: 4.5
index of the maximal element: 2

Continue (y/n)? y

Enter a list of numbers: 2 3 5 6 7 3 4 4.75
maximal element: 7
index of the maximal element: 4

Continue (y/n)? n

Press any key to continue . . . _
```

(3) Based on **Problem P7.7**

• Write the function `reverse(double* a, int a_size())` described in Problem P7.7.

• Write a program that reads in a list of numbers contiguously and saves them in a dynamically allocated array. **Do not use `vectors`** in this problem (see the explanation of how to handle it in the problem 2).

• Let `double* a` be the pointer associated with the array and `int a_size` be the variable containing its size. Apply the function `reverse` to the array located at `a` to reverse the order of its elements. Then display the numbers using the delimiter ", ". When displaying the numbers of the reversed array use a pointer variable, and not an integer index, to traverse the array elements.

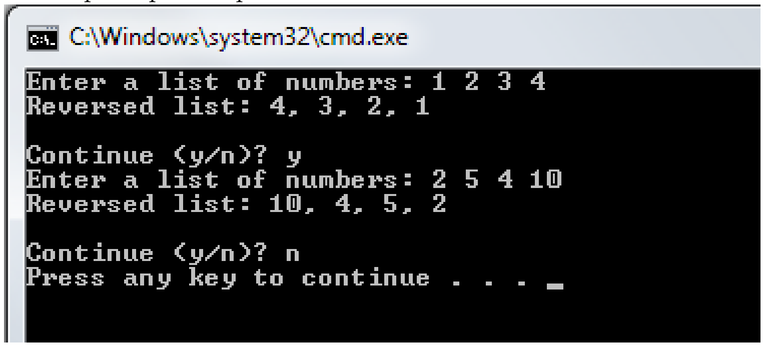
• Implement a loop in which the above actions are repeated until the user requests to quit.

• Assume that the the user's input is always valid.

• Assume that the character after the last number in the input list is the newline character and that there is exactly one space between the numbers in the list.

• Submit the solution in the file named `hmw_1_3.cpp`

- Sample input-output:



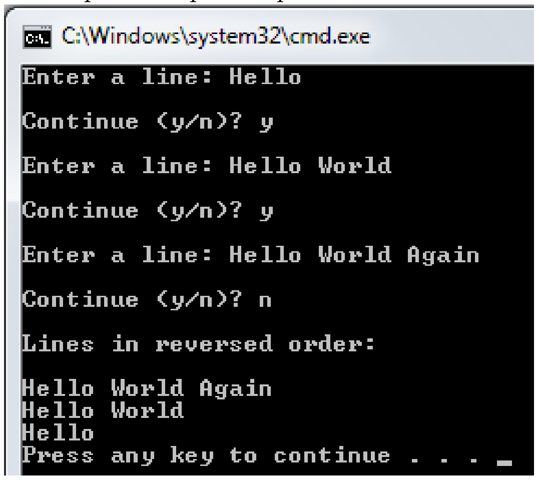
```
C:\Windows\system32\cmd.exe
Enter a list of numbers: 1 2 3 4
Reversed list: 4, 3, 2, 1

Continue (y/n)? y
Enter a list of numbers: 2 5 4 10
Reversed list: 10, 4, 5, 2

Continue (y/n)? n
Press any key to continue . . .
```

(4) **Problem P7.13**.

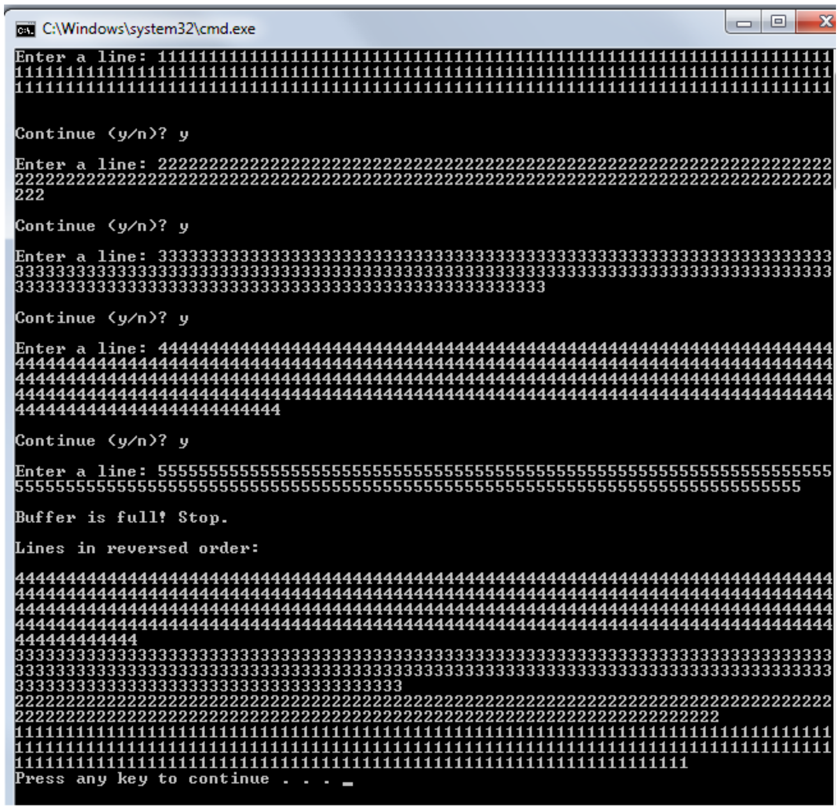
- Implement the program described in **Problem P7.13**.
- Use `getline` to read in the text line into `char buffer[1000]`.
- After a line is read in, ask the user if another line is expected (y/n). If “y”, then read in the next line into `buffer`, otherwise stop the input.
- You can assume that the user inputs no more than 100 lines.
- If a line cannot be placed into `buffer` due to its size, stop the input (see the sample of input-output).
- **Note 1:** `getline` automatically removes the new line character ‘\n’ in the end of the line.
- **Note 2:** To append each line (which is of type `string` as it is read in by using `getline`) to `buffer` use the function `strcpy_s` which deals with C-style strings. The method of the `string` class that returns C-style string, called `c_str()`, will automatically add ‘\0’ to the C-style string.

Submit the solution in the file named `hmw_1_4.cpp` .

- Samples of input-output:



```
C:\Windows\system32\cmd.exe
Enter a line: Hello

Continue (y/n)? y

Enter a line: Hello World

Continue (y/n)? y

Enter a line: Hello World Again

Continue (y/n)? n

Lines in reversed order:

Hello World Again
Hello World
Hello
Press any key to continue . . .
```



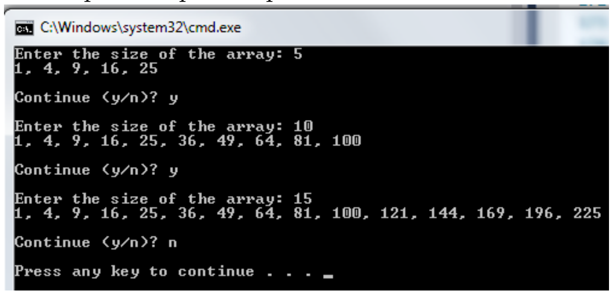
```
Enter a line: 111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111111111111111111


Continue (y/n)? y

Enter a line: 222222222222222222222222222222222222222222222222222222222222222222
22222222222222222222222222222222222222222222222222222222222222222222222222222222
222

Continue (y/n)? y

Enter a line: 333333333333333333333333333333333333333333333333333333333333333333
33333333333333333333333333333333333333333333333333333333333333333333333333333333
3333333333333333333333333333333333333333333333333333

Continue (y/n)? y

Enter a line: 444444444444444444444444444444444444444444444444444444444444444444
44444444444444444444444444444444444444444444444444444444444444444444444444444444
44444444444444444444444444444444444444444444444444444444444444444444444444444444
44444444444444444444444444444444444444444444444444444444444444444444444444444444
44444444444444444444444444

Continue (y/n)? y

Enter a line: 555555555555555555555555555555555555555555555555555555555555555555
55555555555555555555555555555555555555555555555555555555555555555555555555555

Buffer is full! Stop.

Lines in reversed order:

44444444444444444444444444444444444444444444444444444444444444444444444444444444
44444444444444444444444444444444444444444444444444444444444444444444444444444444
44444444444444444444444444444444444444444444444444444444444444444444444444444444
44444444444444444444444444444444444444444444444444444444444444444444444444444444
444444444444
33333333333333333333333333333333333333333333333333333333333333333333333333333333
33333333333333333333333333333333333333333333333333333333333333333333333333333333
33333333333333333333333333333333333333
22222222222222222222222222222222222222222222222222222222222222222222222222222222
222222222222222222222222222222222222222222222222222222222222222222222
11111111111111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111
Press any key to continue . . .
```

(5) Based on **Problem P7.17**

- Implement the function

  `fill_with_values(int a[], int size, intFunPointer f)`

  decribed in **Problem P7.17**.
- Write a program that requests the user to enter a positive integer `int n`, then creates a dynamic array of size n, and then uses the function `fill_with_values` to set the i-th element of the array to `f(i)`, where $i = 0, 1, 2, \dots, n-1$. After that display the result.
- Implement a loop in which the above actions are repeated until the user requests to quit.
- Assume that the the user's input is always valid.
- Submit the solution in the file named `hmw_1_5.cpp`.
- Sample of input-output:



```
Enter the size of the array: 5
1, 4, 9, 16, 25

Continue (y/n)? y

Enter the size of the array: 10
1, 4, 9, 16, 25, 36, 49, 64, 81, 100

Continue (y/n)? y

Enter the size of the array: 15
1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225

Continue (y/n)? n

Press any key to continue . . .
```