CSCE 221 Assignment 4 Cover Page

First Name Jialu Last Name Zhao UIN 426000712

User Name zhaojialu123 E-mail address zhaojialu123@tamu.edu

Please list all sources in the table below including web pages which you used to solve or implement the current homework. If you fail to cite sources you can get a lower number of points or even zero, read more on Aggie Honor System Office website: `http://aggiehonor.tamu.edu/`

Type of sources	web sources
People	
Web pages (provide URL)	https://zh.scribd.com/document/19625974/data-structure-algorithms-pattern-
Printed material	
Other Sources	

I certify that I have listed all the sources that I used to develop the solutions/-codes to the submitted work.
On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work.

Your Name Jialu Zhao Date 03/20/2017

1.Description.

(1) Compile and run the following code, then answer the questions:

(a)What is stored in “`matches`”? 98

(b)What does “`\d`” mean? Digit

```
#include <iostream>
#include <string>
#include <regex>
using namespace std;
int main() {
   regex pattern{R"(\d\d)"};
   string to_search = "I would like the number 98"
               " to be found and printed, thanks.";
   smatch matches;
   regex_search(to_search, matches, pattern);
   for (auto match : matches) {
      cout << match << endl;
   }
   return 0;
}
```

Note that in C/C++ two subsequent strings as above are combined into one string by the compiler.

(c)Modify the regex pattern to retrieve a two-digit number and the word `thanks` in the string. Test your pattern for correctness.

regex pattern{R"((\d\d)([\s\S]+)(thanks))"};

matches[1]=\d\d —search 2 digits

matches[2]=[\s\S]+ — search any other lower case and upper case letters between two digits and “thanks”

matches[3]=thanks — search specific string”thanks”

```
Algorithm:
step 1: set pattern, which has three substrings
step 2: using regex_search to find 3 different substrings
step 3: ouput the substrings that we need
```

input data: given string

ouput data: what we cout

2. Compile and run the following code, then answer the questions:

(a)What does “`\s\S`” mean? lower case and upper case letters

(b)What is stored in `matches[0]`?

<title>This is a title</title>

(c)Why is `matches[1]` different?

This is a title

It’s the substring of matches[0]

```
#include <iostream>
#include <string>
#include <regex>
```

```
using namespace std;
int main() {
    regex pattern{R"(<title>([\s\S]+)</title>)"};
    string to_search = "<html><head>Wow such a header <title>This is a title</title>"
                       "So top</head>Much body</html>";
    smatch matches;
    regex_search(to_search, matches, pattern);
    cout << matches[0] << endl;
    cout << matches[1] << endl;
    return 0;
}
```

(d)Modify the regex pattern to retrieve only the items inside of the header tag but not inside of the title tag. Test your pattern for correctness.

regex pattern{R"((<head>)([\s\S]+)(<title>[\s\S]+</title>)([\s\S]+)(</head>))"};

matches[1]=<head> — search "<head>"

matches[2]=[\s\S]+ —search any other lower case and upper case letters between <head>. and <title>

matches[3]=<title>[\s\S]+</title> –search title

matches[4]=[\s\S]+ — search any other lower case and upper case letters between </title>. and </head>, which is head

```
Algorithm:
step 1: set pattern, which has four substrings
step 2: using regex_search to find 4 different substrings
step 3: ouput the substrings that we need
```

input data: given string

ouput data: what we cout

3. Download the following text file: stroustrup.txt

Write a program using regex that will go through the text file and print out the file name of every hyperlinked powerpoint file. (Hint1: an HTML hyperlink uses the format `<a href=''...>...</a>`. Hint2. The powerpoint file extension is `.ppt`)

regex pattern{R"((<a href=")([\s\S]+.ppt)(">[\s\S]+</a>))"};

matches[1]=<a href= — search <a href=

matches[2]=[\s\S]+.ppt — search the file name that we need

matches[3]=[\s\S]+ —search any other lower case and upper case letters after .ppt which ended with </a>

```
Algorithm:
step 1: read from file
step 2: using while loop to read from file line by line until file's end
step 4: set pattern, which has 5 substrings
step 5: using regex_search to find 5 different substrings
step 6: ouput the substrings that we need if this substring is not empty
```

input data: given string from the given txt file ouput data: what we cout

4. What is the purpose of the functions `regex_search()` and `regex_match()`.

(a) regex_search() looks for its pattern as a substring in the stream

(b) regex_match() looks for a complete match of its pattern and the string

5. Description of data structures and algorithms

(a)We use need a data structure that allows us to keep track, as we go through the string, which characters are legal according to the pattern.

(b)We use a special list structure for this - it contains the possible states in the FSM corresponding to the current and next character in the string being analysed.

(c)This is updated as we simultaneously go through the FSM and move up in the string based on possible state corresponding to current character in string, we can determine from FSM possible states for next character instring.

(d)A variant of the stack/queue is used as theADT: double ended queue allows nodes to be put on front or end of queue: dq.put adds items to end, while dq.push adds items to start.

(d)Split the queue in two halves with specialcharacter

e.g (1 2 + 5 6)

States before "+" represent possible states corresponding to current character.

States after "+" represent possible states corresponding to next character.

6. C++ features or standard library classes

The standard C++ library provides support for regular expressions in the <regex> header through a series of operations. All these operations make use of some typical regex parameters:

(a)Target sequence

(b)Regular expression

(c)Matches array

(d)Replacement string

7.conclusion

(a)Basically, for these questions, we use the same algorithm: separate string to several different substrings and then ouput the substrings that we want.

(b)regex_search() find its pattern as a substring; regex_match() find a complete match of its pattern

(c)The standard C++ library provides support for regular expressions in the <regex>