COSC – Fall 2017
Programming Assignment 1
nfa2dfa – Subset construction to convert NFA to DFA

Your assignment is to implement the subset construction algorithm (Figure 3.32 in the text) to convert an NFA to the equivalent DFA. You will also need to implement other algorithms, such as those for the *move* (Figure 3.31) and ϵ*–closure* (Figure 3.33) operations, required for subset construction.

Both the input NFA and the output DFA will be represented as a transition table. You should represent the NFA and DFA states as integer numbers, starting at 1. A special symbol 'E' will be used to indicate the ϵ–transition. You may choose any language you wish to implement the assignment. The input will come from *standard input* and the output should be printed to *standard output*. You should not make any assumptions regarding the maximum number of states or transitions in the input NFA or the output DFA. You should also carefully consider how you will represent the NFA and corresponding DFA in memory. If you struggle writing the algorithm, consider using a different data structure.

After you complete your implementation, you should rigorously test your program to ensure its correctness. We will test your code using the inputs available from the course website as well as a number of other randomly generated input NFA's.

Finally, you will submit a project report. The project report is a short (one to two page, single-spaced) document that describes:

1. (in your own words) the problem you set out to solve,
2. your approach (i.e. design and relevant implementation details) for solving this problem,
3. how you debugged and tested your solution, and
4. any issues you had in completing the assignment.

Please comment your code so that others (i.e. the grader) can understand it. Comments at the top of the file should indicate your name, this course, and the assignment. You should attempt to match your output as closely as possible to our output.

When you have completed your assignment, you should upload a gzipped tar file (created with `tar cvzf ...`) with your source files and a pdf of your report to the Canvas course website before the end of the day, anywhere on earth, on Friday, September 8^{th}. (That is, you must submit your project by 6:59am EDT on Saturday, September 9^{th} to avoid a late penalty). Partial credits will be given for incomplete efforts. However, a program that does not compile or run will get 0 points. Point breakdown is below:

- Reads / writes from standard input and standard output (10)
- Input file parsed into NFA data structure (10)
- NFA to DFA conversion (40)
- Output is clear and in the appropriate format (10)
- Code is easy to understand (e.g., contains appropriate comments) (10)
- Project report (20)

Example input NFA:

```
Initial State:  1
Final States:   {11}
Total States:   11
State   a               b               E
1       {}              {}              {2,5}
2       {3}             {}              {}
3       {}              {4}             {}
4       {}              {}              {8}
5       {}              {6}             {}
6       {7}             {}              {}
7       {}              {}              {8}
8       {}              {}              {9,11}
9       {10}            {}              {}
10      {}              {}              {9,11}
11      {}              {}              {}
```

Output from NFA2DFA executable:

```
reading NFA ... done.

creating corresponding DFA ...
new DFA state:  1    -->  {1,2,5}
new DFA state:  2    -->  {3}
new DFA state:  3    -->  {6}
new DFA state:  4    -->  {4,8,9,11}
new DFA state:  5    -->  {7,8,9,11}
new DFA state:  6    -->  {9,10,11}
done.

final DFA:
Initial State:  1
Final States:   {4,5,6}
Total States:   6
State   a               b
1       {2}             {3}
2       {}              {4}
3       {5}             {}
4       {6}             {}
5       {6}             {}
6       {6}             {}
```