Pre-lab Part 1
1.
```
while (fscan %c%c\n != E0F){
        check if its alphabet;
        check if its uppercase;
        make it into its respective number depending on the options
        matrix[row][col] = 1;
        if undirected{
                matrix [col][row] = 1;
        }
}
```
2. A -> B, B -> C/D, C -> F/Z, D -> E, F -> Z
3. The worst case complexity of the post order traversal tree would be it going through the whole tree and not finding an answer, and that happening until the end. In general we would say that the worst case complexity would be finding it in 2^n, n being the number of inputs. Using the assignment, for example, the worst case complexity would going be to go down the whole alphabet for you to find the solution, or really late into the algorithm.

Pre-lab Part 2
1.
```
Stack create{
        stack malloc size of stack
        capacity = minimum
        stack->top = 0;
        stack->items = calloc(minimum, size of items);
        return stack;
}
Stack delete {
        free (stack->items);
}
Stack empty {
        check if top == 0;
}
Stack size {
        return top of stack = size;
}
Stack push {
        checks if stack top = capacity
                if it is, add more stack space
        stack->items top = item
        increment
}
Stack pop {
        check if its empty first
        if not, take one out from top and decrement top
}
```

Stack print: prints from index 0 to top

MAIN:
- uses switch case for all the options
- default for *fp is stdin, unless user inputs file
- make matrix of size n x n, where n is number of junctions, in this case it is vertices = 26

read_file:
- this is to read the file given, whether it is user inputted or given file
- parameters is a file pointer and the matrix
- scan line by line with scanf, checking if it is uppercase and in the alphabet, since you need to subtract different values to make the char to int, depending on case and if it is in alphabet
- once converted, do matrix[row][col]
- if user wants undirected, do matrix[col][row] too