Assignment 1 Clarifications

- Q1b: State whether having the code snippet in your code will cause compilation errors. Explain why or why not.

- Q3: For the code snippets, how many times is the printf statement executed? Briefly explain (up to 3 sentences).

- Q4: Write a function to draw an isosceles triangle of a given height N using for/while loop.

- Hand in hard copy before Wednesday class in CSIL (across from main CS office)
Exam Hardship

- Three or more finals scheduled within 24 hours
  - Let me know via email ASAP if you haven’t already
  - mochen@cs.sfu.ca

- Exam at one location (e.g. Burnaby) followed immediately by exam at another location (e.g. Surrey).
  - Math 152 Final will be in Burnaby
  - Please check with and thank Prof. Jungic
    - Please keep up with both courses
    - Last two lectures will be review sessions
Array Comparison, Strings and Loops

CMPT 125
Jan. 14
Lecture 5

Today

- Array Comparison
- Strings
- Nested Loops
Array Comparison

Puzzle: What’s wrong with this code?

```c
int main() {
    int password[3] = {1,2,3};
    int answer[3];

    for (int i = 0; i < 3; i++) {
        printf("Enter digit %d: ", i+1);
        scanf("%d", answer+i);
    }

    if (password != answer) {
        printf("Incorrect password!\n");
    }
}
```

**probably a bug**

compares the values of the pointers, not the array elements
Array Comparison

- Write a function to compare two arrays
- Array parameters passed by base address
  - Style points: use int arr[] instead of int *arr

```c
int arrCompare(int A[], int B[], int length) {
    for (int i = 0; i < length; i++) {
        if (A[i] < B[i]) {
            return -1;
        } else if (A[i] > B[i]) {
            return 1;
        }
    }
    return 0;
}
```
Arrays of char

- **type** `char` is 1 byte per element
  - traditionally to hold one ASCII character
  - an array of `char` is a string!
- **end of string terminated by null char: `'\0'`**

```c
int main ( ) {
    char msg[10] = "ur n00b!";
    printf("%s\n", msg);
}
```

<table>
<thead>
<tr>
<th></th>
<th>[0]</th>
<th>[1]</th>
<th>[2]</th>
<th>[3]</th>
<th>[4]</th>
<th>[5]</th>
<th>[6]</th>
<th>[7]</th>
<th>[8]</th>
<th>[9]</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg[10]:</td>
<td>'u'</td>
<td>'r'</td>
<td>' '</td>
<td>'n'</td>
<td>'0'</td>
<td>'0'</td>
<td>'b'</td>
<td>'!'</td>
<td>'0'</td>
<td></td>
</tr>
<tr>
<td></td>
<td>117</td>
<td>114</td>
<td>32</td>
<td>110</td>
<td>48</td>
<td>48</td>
<td>98</td>
<td>33</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Dec</td>
<td>Hx Oct</td>
<td>Char</td>
<td>Dec</td>
<td>Hx Oct</td>
<td>Html</td>
<td>Char</td>
<td>Dec</td>
<td>Hx Oct</td>
<td>Html</td>
<td>Char</td>
</tr>
<tr>
<td>-----</td>
<td>--------</td>
<td>----------</td>
<td>-----</td>
<td>--------</td>
<td>------</td>
<td>----------</td>
<td>-----</td>
<td>--------</td>
<td>------</td>
<td>----------</td>
</tr>
<tr>
<td>0</td>
<td>000</td>
<td>NUL (null)</td>
<td>32</td>
<td>20 040</td>
<td> </td>
<td>Space</td>
<td>64</td>
<td>40 100</td>
<td>@</td>
<td>@</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>SOH (start of heading)</td>
<td>33</td>
<td>21 041</td>
<td>!</td>
<td>!</td>
<td>65</td>
<td>41 101</td>
<td>&amp; #65;</td>
<td>A</td>
</tr>
<tr>
<td>2</td>
<td>002</td>
<td>STX (start of text)</td>
<td>34</td>
<td>22 042</td>
<td>“</td>
<td>&quot;</td>
<td>66</td>
<td>42 102</td>
<td>&amp; #66;</td>
<td>B</td>
</tr>
<tr>
<td>3</td>
<td>003</td>
<td>ETX (end of text)</td>
<td>35</td>
<td>23 043</td>
<td>#</td>
<td>#</td>
<td>67</td>
<td>43 103</td>
<td>&amp; #67;</td>
<td>C</td>
</tr>
<tr>
<td>4</td>
<td>004</td>
<td>EOT (end of transmission)</td>
<td>36</td>
<td>24 044</td>
<td>$</td>
<td>$</td>
<td>68</td>
<td>44 104</td>
<td>&amp; #68;</td>
<td>D</td>
</tr>
<tr>
<td>5</td>
<td>005</td>
<td>ENQ (enquiry)</td>
<td>37</td>
<td>25 045</td>
<td>%</td>
<td>%</td>
<td>69</td>
<td>45 105</td>
<td>&amp; #69;</td>
<td>E</td>
</tr>
<tr>
<td>6</td>
<td>006</td>
<td>ACK (acknowledge)</td>
<td>38</td>
<td>26 046</td>
<td>&amp;</td>
<td>&amp;</td>
<td>70</td>
<td>46 106</td>
<td>&amp; #70;</td>
<td>F</td>
</tr>
<tr>
<td>7</td>
<td>007</td>
<td>BEL (bell)</td>
<td>39</td>
<td>27 047</td>
<td>’</td>
<td>’</td>
<td>71</td>
<td>47 107</td>
<td>&amp; #71;</td>
<td>G</td>
</tr>
<tr>
<td>8</td>
<td>010</td>
<td>BS (backspace)</td>
<td>40</td>
<td>28 050</td>
<td>(</td>
<td>(</td>
<td>72</td>
<td>48 110</td>
<td>&amp; #72;</td>
<td>H</td>
</tr>
<tr>
<td>9</td>
<td>011</td>
<td>TAB (horizontal tab)</td>
<td>41</td>
<td>29 051</td>
<td>)</td>
<td>)</td>
<td>73</td>
<td>49 111</td>
<td>&amp; #73;</td>
<td>I</td>
</tr>
<tr>
<td>A</td>
<td>012</td>
<td>LF (NL line feed, new line)</td>
<td>42</td>
<td>2A 052</td>
<td>*</td>
<td>*</td>
<td>74</td>
<td>4A 112</td>
<td>&amp; #74;</td>
<td>J</td>
</tr>
<tr>
<td>B</td>
<td>013</td>
<td>VT (vertical tab)</td>
<td>43</td>
<td>2B 053</td>
<td>+</td>
<td>+</td>
<td>75</td>
<td>4B 113</td>
<td>&amp; #75;</td>
<td>K</td>
</tr>
<tr>
<td>C</td>
<td>014</td>
<td>FF (NP form feed, new page)</td>
<td>44</td>
<td>2C 054</td>
<td>-</td>
<td>-</td>
<td>76</td>
<td>4C 114</td>
<td>&amp; #76;</td>
<td>L</td>
</tr>
<tr>
<td>D</td>
<td>015</td>
<td>CR (carriage return)</td>
<td>45</td>
<td>2D 055</td>
<td>0</td>
<td>0</td>
<td>77</td>
<td>4D 115</td>
<td>&amp; #77;</td>
<td>M</td>
</tr>
<tr>
<td>E</td>
<td>016</td>
<td>SO (shift out)</td>
<td>46</td>
<td>2E 056</td>
<td>1</td>
<td>1</td>
<td>78</td>
<td>4E 116</td>
<td>&amp; #78;</td>
<td>N</td>
</tr>
<tr>
<td>F</td>
<td>017</td>
<td>SI (shift in)</td>
<td>47</td>
<td>2F 057</td>
<td>2</td>
<td>2</td>
<td>79</td>
<td>4F 117</td>
<td>&amp; #79;</td>
<td>O</td>
</tr>
<tr>
<td>10</td>
<td>020</td>
<td>DLE (data link escape)</td>
<td>48</td>
<td>30 060</td>
<td>3</td>
<td>3</td>
<td>80</td>
<td>50 120</td>
<td>&amp; #80;</td>
<td>P</td>
</tr>
<tr>
<td>11</td>
<td>021</td>
<td>DC1 (device control 1)</td>
<td>49</td>
<td>31 061</td>
<td>4</td>
<td>4</td>
<td>81</td>
<td>51 121</td>
<td>&amp; #81;</td>
<td>Q</td>
</tr>
<tr>
<td>12</td>
<td>022</td>
<td>DC2 (device control 2)</td>
<td>50</td>
<td>32 062</td>
<td>5</td>
<td>5</td>
<td>82</td>
<td>52 122</td>
<td>&amp; #82;</td>
<td>R</td>
</tr>
<tr>
<td>13</td>
<td>023</td>
<td>DC3 (device control 3)</td>
<td>51</td>
<td>33 063</td>
<td>6</td>
<td>6</td>
<td>83</td>
<td>53 123</td>
<td>&amp; #83;</td>
<td>S</td>
</tr>
<tr>
<td>14</td>
<td>024</td>
<td>DC4 (device control 4)</td>
<td>52</td>
<td>34 064</td>
<td>7</td>
<td>7</td>
<td>84</td>
<td>54 124</td>
<td>&amp; #84;</td>
<td>T</td>
</tr>
<tr>
<td>15</td>
<td>025</td>
<td>NAK (negative acknowledge)</td>
<td>53</td>
<td>35 065</td>
<td>8</td>
<td>8</td>
<td>85</td>
<td>55 125</td>
<td>&amp; #85;</td>
<td>U</td>
</tr>
<tr>
<td>16</td>
<td>026</td>
<td>SYN (synchronous idle)</td>
<td>54</td>
<td>36 066</td>
<td>9</td>
<td>9</td>
<td>86</td>
<td>56 126</td>
<td>&amp; #86;</td>
<td>V</td>
</tr>
<tr>
<td>17</td>
<td>027</td>
<td>ETB (end of trans. block)</td>
<td>55</td>
<td>37 067</td>
<td>:</td>
<td>:</td>
<td>87</td>
<td>57 127</td>
<td>&amp; #87;</td>
<td>W</td>
</tr>
<tr>
<td>18</td>
<td>030</td>
<td>CAN (cancel)</td>
<td>56</td>
<td>38 070</td>
<td>;</td>
<td>;</td>
<td>88</td>
<td>58 130</td>
<td>&amp; #88;</td>
<td>X</td>
</tr>
<tr>
<td>19</td>
<td>031</td>
<td>EM (end of medium)</td>
<td>57</td>
<td>39 071</td>
<td>&lt;</td>
<td>&lt;</td>
<td>89</td>
<td>59 131</td>
<td>&amp; #89;</td>
<td>Y</td>
</tr>
<tr>
<td>20</td>
<td>032</td>
<td>SUB (substitute)</td>
<td>58</td>
<td>3A 072</td>
<td>&gt;</td>
<td>&gt;</td>
<td>90</td>
<td>5A 132</td>
<td>&amp; #90;</td>
<td>Z</td>
</tr>
<tr>
<td>21</td>
<td>033</td>
<td>ESC (escape)</td>
<td>59</td>
<td>3B 073</td>
<td>\</td>
<td>;</td>
<td>91</td>
<td>5B 133</td>
<td>&amp; #91;</td>
<td>{{</td>
</tr>
<tr>
<td>22</td>
<td>034</td>
<td>FS (file separator)</td>
<td>60</td>
<td>3C 074</td>
<td>^</td>
<td>^</td>
<td>92</td>
<td>5C 134</td>
<td>&amp; #92;</td>
<td>\</td>
</tr>
<tr>
<td>23</td>
<td>035</td>
<td>GS (group separator)</td>
<td>61</td>
<td>3D 075</td>
<td>_</td>
<td>_</td>
<td>93</td>
<td>5D 135</td>
<td>&amp; #93;</td>
<td>{ }</td>
</tr>
<tr>
<td>24</td>
<td>036</td>
<td>RS (record separator)</td>
<td>62</td>
<td>3E 076</td>
<td>`</td>
<td>`</td>
<td>94</td>
<td>5E 136</td>
<td>&amp; #94;</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>037</td>
<td>US (unit separator)</td>
<td>63</td>
<td>3F 077</td>
<td>DEL</td>
<td>DEL</td>
<td>95</td>
<td>5F 137</td>
<td>&amp; #95;</td>
<td>_</td>
</tr>
</tbody>
</table>

Source: www.LookupTables.com
#include <stdio.h>
#include <string.h>

int main ( ) {
    char password[4] = "abc";
    char answer[4];

    printf("Enter 3-character code: ");
    scanf("%s", answer);
    if (strcmp(password, answer) != 0) {
        printf("Incorrect password!\n");
    }
}

C library function to do string comparisons:
- 0 means equal
- < 0 means first < last
- > 0 means first > last

not &answer because answer is a pointer!
Common String Functions

int strlen(char s[])  
- returns the length of the string
- counts characters until null terminator
- Q: What happens if there is no null terminator?

void strcpy(char dest[], char src[])  
- copies the string dest[] ← src[]
- Q: What must be true about dest[]?
String I/O

Input

- `scanf("%s", str);`
- `scanf("%[^\n]s", str);`
- `gets(str);`

Output

- `printf("%s", str);`
- `puts(str);`

Also dangerous

Dangerous
Nested Loops

- It is possible to include any sequence of statements within a loop body including:
  - calculations
  - function calls
  - if statements
  - other loops
- Just like you did in Python!

Classic Problem: Write a function that scans an array of int. It returns 1 if and only if two of the elements are the same, 0 otherwise.
int dup_chk(int a[], int length) {
    int i = length;
    while (i > 0) {
        i--;
        int j = i - 1;
        while (j >= 0) {
            if (a[i] == a[j]) {
                return 1;
            }
            j--;
        }
    }
    return 0;
}

Simulation:
dup_chk(a[4], 4):

    j       j       j       j       j       i
a[4]:  5 3 9 4

These statements run the most frequently in the worst case
- What is the worst case?
- How many times when length = 4?
Another Performance Measure

● Often consider the worst-case behaviour as a benchmark
  ○ make guarantees about code performance under all circumstances

● Can predict performance by counting the number of steps required by algorithm in the worst case
  ○ Derive total steps (T) as a function of input size (N)
Q. What is $N$?
- The number of elements in the array

```c
int dup_chk(int a[], int length) {  
  int i = length;  

  while (i > 0) {  
    i--;  
    int j = i - 1;
    while (j >= 0) {  
      if (a[i] == a[j]) {  
        return 1;  
      }
      j--;  
    }
    i--;  
  }

  return 0;
}
```

Outside of loop: 2 (steps)

Outer loop: $3N + 1$

Inner loop: $3i + 1$ for all possible $i$ from 0 to $N - 1$.

$$= \frac{3}{2} N^2 - \frac{1}{2} N$$

Grand total = $\frac{3}{2} N^2 + \frac{5}{2} N + 3$

A quadratic function!
Empirical Measurement

- Another graph - a quadratic this time!
- Confirms predictions: *doubling* (x2) the input size leads to *quadrupling* (x4) the running time

<table>
<thead>
<tr>
<th>N</th>
<th>time (in ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10,000</td>
<td>89</td>
</tr>
<tr>
<td>20,000</td>
<td>365</td>
</tr>
<tr>
<td>40,000</td>
<td>1,424</td>
</tr>
<tr>
<td>100,000</td>
<td>9,011</td>
</tr>
</tbody>
</table>
2D Maximum Density Problem

Problem: Given a 2-dimensional array \((N \times N)\) of integers, find the 10x10 patch that yields the largest sum.

Applications:
- Resource management and optimization
- Finding brightest areas of photos
Algorithm / Code?

- Simple approach: Try all possible positions for the upper left corner
  - $(N-9) \times (N-9)$ of them
  - use a nested loop
- add each patch using a 10x10 nested loop
- A brute-force approach!
  - Generate a possible solution [naively]
  - Test it [naively]
In C

```c
int max10by10(int a[N][N]) {
    int best = 0;
    int total = 0;
    for (int row = u_row; row < u_row+10; row++) {
        for (int col = u_col; col < u_col+10; col++) {
            total += a[row][col];
        }
    }
    best = max(best, total);
    return best;
}
```
In C

```c
int max10by10(int a[N][N]) {
    int best = 0;
    for (int u_row = 0; u_row < N-9; u_row++) {
        for (int u_col = 0; u_col < N-9; u_col++) {
            int total = 0;
            for (int row = u_row; row < u_row+10; row++) {
                for (int col = u_col; col < u_col+10; col++) {
                    total += a[row][col];
                }
            }
            best = max(best, total);
        }
    }
    return best;
}
```

Approximate Method:

Count the *barometer instructions*, the instructions executed most frequently. Usually, in the innermost loop.

Innermost loop: $11 + 10 + 10 = 31$ ops

Total = $31 \times 10 \times (N-9) \times (N-9) = 310N^2$