Tutorial Sheet (August 17, 2018)

ESC101 – Fundamentals of Computing

Announcements

1. **Extra lecture on Saturday, August 18, 2018, L20, 12noon**
2. **Extra lab for B10, 11, 12, 14, August 18, 2018, CC-01,CC-02, 2PM**
3. Students are advised to enrol on Piazza if not already done so. In case of login troubles, please contact instructor. If not on Piazza, students will miss discussions, tips, announcements.
4. Students should attempt practice problems (new ones as well as lab problems released as practice problems) on a weekly basis.
5. Students are advised not to wait till exam time to practice.
6. Students should come to lab on time, remember CC password.

Demonstrations

1. Please warn students that Prutor shows them, in the evaluation tab, results of the current as well as previous evaluations/submits. Always scroll (using mouse wheel or scroll bar) to the top of that frame to view results of most recent evaluation. Students sometimes keep looking at older evaluations and get confused why their code is still incorrect.
2. Sometimes when there is too much text in output, the evaluation tab does not format things nicely. Decreasing font size on the browser window is a good way to get a nice formatting. Please show how to increase/decrease font size on browser (on Firefox CTRL+- decreases and CTRL++ increases font size).
3. Please demonstrate how to check in Prutor if there are extra trailing spaces at the end of a line in their output. Click on Evaluate/Submit and then select using mouse, both the expected output and the actual output. Upon selection, any extra spaces get revealed.

4. Please demonstrate the use of simple but timesaving and very useful keyboard shortcuts on Prutor.

a. CTRL+S: save
b. CTRL+C: copy
c. CTRL+V: paste
d. CTRL+Z: undo
e. CTRL+Y: redo
f. CTRL+SHIFT+C: compile
g. CTRL+SHIFT+X: execute
h. CTRL+SHIFT+V: evaluate

Revision (ask for doubts)

1. Mixed-type expressions (int * float, int * long etc)
2. Use of format specifiers in floats %f vs %0.1f and %e vs %0.2e
3. math.h and various functions therein abs, fabs, sqrt, floor, ceil
4. Statements vs expressions

   c = a + b is an expression
   a + b is also an expression
   a is also an expression
   a + is not valid expression

   c = is not a valid expression
   c = a + b; is a statement
   printf("%d",a); is a statement
   a; is a statement

5. Expressions are evaluated according to the BODMAS rule
   Two new rules: unary negation at the top (above brackets) and assignment (below everything else).
6. Mathematical expressions generate numerical values
   int a, b = 2, c = 3;
   b + 7 will generate value 9
   b + c will generate value 5
   b will generate value 2
   a = 2 * b will generate value 4

   printf("%d",a = 2 * b); will print 4
   This is how expressions like p = q = r = 7 are evaluated.
7. if condition with single clause condition if (a < 10) if (b > 20) etc

---

Sample Questions to discuss

Evaluation order of the expression x = - 5 * 4 / 2 * 3 + - 1 * 2 – 6 * 3

1. **Highest**: unary negation
   x = **(**- 5**)** * 4 / 2 * 3 + **(**- 1**)** * 2 – 6 * 3
2. **Next highest**: multiply, division, among them, left associativity
   x = **(((**(- 5) * 4**)** / 2**)** * 3**)** + **(**(- 1) * 2**)** – **(**6 * 3**)**
3. **Next highest**: add, subtraction, among them, left associativity
   x = **((**((((- 5) * 4) / 2) * 3) + ((- 1) * 2)**)** – (6 * 3)**)**
4. **Lowest**: assignment
   (x = ((((((- 5) * 4) / 2) * 3) + ((- 1) * 2)) – (6 * 3))**)**

Compute the factorial of 20 (does not fit inside int but fits inside long).

```
#include<stdio.h>
int main(){
    long prod = 20*19*18*17*16*15*14*13*12*11*10*9*8*7*6*5*4*3*2*1;
    printf("%ld", prod);
    return 0;
}
```

The above program does not work (integer multiplication going on). Need to typecast explicitly. However, doing so in the wrong manner

```
prod = 20*19*18*17*16*15*14*13*12*11*10*9*8*7*6*5*4*3*2*(long)1;
```

also does not work ☹ due to associativity rules. Left to right bracketing and so first 20*19, 380*18 etc. calculations are done as integers. By the time the typecast (long)1 is encountered, damage already done.

```
prod = (long)20*19*18*17*16*15*14*13*12*11*10*9*8*7*6*5*4*3*2*1;
```

works since the very first sub-expression to be evaluated is (long)20*19 which produces a long due to automatic typecasting. Thereafter, Mr C automatically typecasts everything else to long.

Some Pitfalls and recognizing compiler error messages

1. Typecasting floats to int will cause loss of digits after decimal
   float a = 2.5; int b = (int)a; // b = 2
2. Typecasting int to float or long to float may also cause distortion
   long c = 9765432123; a = (float)c; // 9765432320.000000
3. Rounding-errors in integer division can accumulate
4. (a + b)/2 vs a/2 + b/2
5. Only ( ) are valid brackets in math expressions/formulae. [ ] and { } are not valid brackets for math expressions.
6. Not properly bracketing if condition statements may cause unexpected results.
