Australian National University

Computer Science Courses
ANU College of Engineering & Computer Science

Week 05

Students are expected to attempt ALL of the questions in the Multi choice, Short answer and Lab quesitons. They will be discussed and marked at the beginning of the lab.

Multi choice

Question: How many times is printf('a') executed in the below?

```
for (i=0;i<4;i++) {
    for (j=i;j<4;j++) {
        printf('a');
     }
}
```

A) 16
B) 4
C) 10
D) 17

C

4+3+2+1

Question: Consider the below macro:

```
#define ADDTWO(x) 2 + x
```

What would the expression 2*ADDTWO(3) return?

A) 7
B) 8
C) 9
D) 10

A

Macros are just simple textual replacement

Question: What would the below program print?

```
#include<stdio.h>
void fun(int x) {
    x = 6;
}
int main() {
    int y;
    y = 5;
    fun(y);
    printf("%d",y);
```

A) 4
B) 5
C) 6
D) 7

B

Argument passing mechnism

```
    }
```

Question: In c the length of a string is determined by _________ D low-level implementation of string	**A)** the length of the array that holds the string. **B)** an integer which stores the length of the string. **C)** -1 which is the end marker of the string. **D)** a 0 (sometimes referred to as a null) which marks the end of the sequence of characters.

Short answer

Question: 'sysvbanner' is a simple unix program that enables you to turn short strings into a large ascii banners.

```
$ sysvbanner amuse

  ##    #    #  #    #   ####   ######
 #  #   ##  ##  #    #  #       #
#    #  # ## #  #    #   ####   #####
######  #    #  #    #       #  #
#    #  #    #  #    #  #    #  #
#    #  #    #   ####    ####   ######
```

It's quite hard...

If you were asked to design and implement this program in c what approach would you use? What are the complex aspects of this implementation? How would you overcome them. What approach would you use to store font information? How general would this approach be? Would it work with different sized fonts? How much memory does your approach require? Is this minimal? Note that, in this tutorial question you do not need to implement the banner program, you just need to think about how it would get implemented.
Question: What is the relationship between robustness, readability, and performance in c programming?

Lab questions

Excercise: Rock, paper, sizers, is a simple game between two opponents. Each opponent selects between one of rock, paper, or sizers at the same time as each other. Rock wins over sizers, sizers wins over paper, paper wins over rock. If the opponents selected the same item then it is a draw. Write a text based c program that lets you plays rock, paper, sizers with the computer. The item the user selects is given via standard input with the key press 'r' for rock, 'p' for paper, and 's' for sizers. The program should enable a series of games to be played and keep track of games won. Note the computer shouldn't cheat, that is it should select its item before it processes the one given by the user. A simple strategy is to just use random number for selecting the item (see 'man 3 rand'). A typical interaction someone may have with this program is:

```
Rock, Paper, Sizers.  Enter one of: 'r', 'p', 's'. % r
You selected: Rock      Computer selected: Sizers
You win. :(
Your score: 1          Computer : 0
Rock, Paper, Sizers.  Enter one of: 'r', 'p', 's'. % p
You selected: Paper      Computer selected: Paper
Draw.
Your score: 1          Computer : 0
Rock, Paper, Sizers.  Enter one of: 'r', 'p', 's'. % r
You selected: Rock      Computer selected: Paper
Computer Wins. :)
Your score: 1          Computer : 1
Rock, Paper, Sizers.  Enter one of: 'r', 'p', 's'. %
```

(just think about this one) Would it be possible to write a program that could win more often than not? Could you develop a program that could eventually win over the program you implemented?

In-class group task

This exercise can be done in groups of two or three. Your tutor will give you a programming exercise that involves drawing shapes with ascii. Attempt to make your code 'robust'. Get another group in your class to run your program and see if they can break it. Also do a 'code review' of each others code. As the lab group discuss what tests people did to check for robustness. Also talk about how their code could be improved and if anything came of the code reviews.