16.070 EXAM REVIEW GUIDE

EXAM ON 3/4/02, Monday, in the WALKER GYM - 50-340

As a student of 16.070 you are responsible for the following material:

Chapters covered: CH 1-4, CH 5.1-5.3; 5.6-5.7	**Lecture material covered:** Up to and including Lecture # 8	**Recitation material covered:** Up to and including week 3
Problem Sets 1-3	**Labs** Through week 4	**Style Guide and muddy points:** All of it

General tips:

Know thy variable scope

Variables can either be local or global. Understand the difference between the two, where each of these types are declared, and peculiarities in their use. Also, understand parameter passing between functions.

Honor the ancient practice of good style

The most important thing in writing code down is proper documentation and readability. Unreadable clever code that works now is worse than inefficient but readable and understandable code. This is especially true when writing out short snippets of code. Proper commenting may not always be possible, but it is always possible to make the code more readable by proper indentation, good variable names, etc.

Thou shalt not forsake modularity

Monolithic code is neither easy to read nor easy to debug. Functions add flexibility in at least two ways: it allows the programmer to easily reuse a piece of code, and it also allows the programmer to break up the function into separate pieces. The major trick to getting functions to work correctly is to perfect the fine art of parameter passing.

Switches shall light thy path

Switch statements are most useful when one variable has several states. Used in conjunction with the case statement, it saves the programmer from making an undue amount of if statements. Properly implementing a case statement is a prerequisite for more difficult state transition diagrams, because is allows for generally cleaner code.

Looping shall set thee free?

Always understand the loop well enough to break out of it. For and While loops each have their own idiosyncrasies in usage. Proper initialization of variables is paramount to the proper processing of the loop. Loops are often the heart of the program.

Const thou #define, nay declare! Life?

Variables are the lifeblood. Understand the difference between the const statement, as well as the #define preprocessor directive. Know not only the different types of variables, but also how to represent them in a printf or related statement.

If ((time == NULL) && (frustration > expected))
Printf(" I must be in 16.070") ;
Else
Printf(" wake me for the apocalypse");

Logical operators are the brains of the function. The IF statement allows the program to decide a future course of action based upon the relational operators. Knowing how to use the IF and ELSE commands with the proper operators is crucial to the more complex programs.

Thou shalt not steal, but if thy must, utilize fclose() to cover thy tracks

File operations are useful for handling large amounts of data. There are three main phases: file opening, data storage or retrieval, and file closing. Look over and understand the various commands such as fscanf, fputc, etc. and their usage.

Honor the zero index and keep it h0ly.

Array counting starts with zero, and runs up to size – 1. The declaration of an array is denoted by square brackets like this: int array[size];. Accessing arrays requires inputting an integer number into the square brackets after the array variable name. Arrays are powerful tools useful for storage of large amounts of information of the same data type.

<u>Helper problems</u>

1) Run through the first 4 steps of the software design process for a sensor that determines if the main landing gear of a plane has touched down on the runway.

2) This short section of code has problems. Search and destroy the errors

```
        #define PI 3;
        Int count=0;

int main(void)
{
        While (count >=0)
                printf("I can count!  See…./n")
                printf( "%f", num);
                printf(" /nI told you I could count!");
                printf(" can we have pumpkin %d  now?" , PI);
                if (count = 5 )
                count = -50;
}
```

3) The _________ translates your code into ______ language so that your computer can execute the code. However, if you have a ____ in your code it can prevent the program from executing the algorithm correctly, despite the total lack of warning and _____ messages.

4) Perform a program trace (memory snapshots) of the six variables below, and determine the value of total when it prints to the screen.

```
        #include <stdio.h>

int main(void)
{
        double F14=25.0;
        double F16=50.0;
        double SU30=25.0;
        int SAM=0;
        double total=0.0;
        double pK=0.9;

        total=F14+F16+SU30;
        for (SAM=0;SAM<4;SAM++)
        {
                F14 =F14 - (1-pK)* F14 *SAM;
                F16 =F16 - (1-pK)* F16 *SAM *(1-pK);
                SU30=SU30- (1-pK)* SU30 *SAM;
        }

        total=F14+F16+SU30;

        printf("%f\n",total);
        printf("%f\n",F14);

}
```

5) Write a function that will input the first 1000 digits after the decimal of pi from a file called “pi.dat”. The file will start off 14159…

6) Evaluate by hand every printf statement in the following program:

```
#include <stdio.h>
int main(void)
{
        int count1=0;
        int count2=0;


        for (count1=0;count1<=4;count1++)
        {
                printf("%d: ", count1);
                for (count2=count1;count2>=0;count2--)
                {
                        printf("%d",count2);
                }
                printf("\n");
        }
}
```

7) Give a short definition of the following terms

Argument:

Array:

Binary:

Debug:

End-of-file:

Parameter:

Preprocessor Directives:

Relational Operator:

Scope:

Simulation:

Source Program: