Code Examples from Lectures on 7-3_functions

callByValue.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/7-3_functions/code/callByValue.c)

Simple example illustrating call by value

```
#include <stdio.h>
#include <stdlib.h>

void f(int n);

int main(void) {
    int  n = 5;

    f(n);

    printf("n=%d\n", n);
    return EXIT_SUCCESS;
}

void f(int n){
    n = 42;
}
```

answerEx.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/7-3_functions/code/answerEx.c)

Demonstration of a user defined function

Exercise: We took answer.c and modified it to take user input until the user enters non-integer data

```
#include <stdio.h>
#include <stdlib.h>
//function prototypes
double answer(int x);
int addNumbers(int num1, int num2);

int main(void) {
    int value;
    int value2,value3;

    printf("Please enter 3 integer values: ");
    while(scanf("%d %d %d",&value,&value2,&value3) == 3){

        printf("The answer is %lf\n",answer(value));
        printf("The sum is %d\n",addNumbers(value2,value3));
        printf("Please enter 3 integer value: ");
    }
    return EXIT_SUCCESS;
}

//function definition/implementation
double answer(int x){
    return x * 0.5;
}

int addNumbers(int num1, int num2){
    int sum;
    sum = num1 + num2;
    return sum;
}
```

answer.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/7-3_functions/code/answer.c)

Demonstration of a user defined function

```c
#include <stdio.h>

//function prototype
int answer(double x);

int main(void) {

    //Calling (using the function);
    printf("answer(2) = %d\n",answer(2));

    return 0;
}

//function definition/implementation
int answer(double x){
    return x * 42;
}
```

lec_mypow.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/7-3_functions/code/lec_mypow.c)

```c
#include <stdio.h>

int power(int, int);

int main(void) {
    int base, exponent, result1, result2;

    printf("Base: ");
    scanf("%d", &base);
    printf("Exponent: ");
    scanf("%d", &exponent);

    result1 = power(base, exponent);
    result2 = power(exponent, base);
    printf("Result1: %d\n", result1);
    printf("Result2: %d\n", result2);

    return 0;
}

int power(int base, int exponent) {
    int i = 0;
    int product = 1;
    while (i < exponent) {
        product = product * base;
        i = i + 1;
    }
    return product;
}
```

lec_func2.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/7-3_functions/code/lec_func2.c)

```c
#include <stdio.h>

void printPlusOne(int);
int plusOne(int);

int main(void) {
    int number;
    number = 4;
    printf("main number: %d\n", number);
    printPlusOne(number);
    printf("fun2 number: %d\n", plusOne(number));

    return 0;
}

void printPlusOne(int passedInNumber) {
    printf("fun number: %d\n", passedInNumber + 1);
}

int plusOne(int passedInNumber) {
    return passedInNumber + 1;
}
```

lec_func6.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/7-3_functions/code/lec_func6.c)

```c
#include <stdio.h>

#define TRUE 1
#define FALSE 0

int biggerThanFive(int);

int i = 5;

int main(void) {
    int number;
    scanf("%d", &number);
    printf("%d\n", biggerThanFive(number));
    printf("i main %d\n", i);
    return 0;
}

int biggerThanFive(int n) {
    int answer;
    if (n > 5) {
        answer = TRUE;
    } else {
        answer = FALSE;
    }
    printf("Done!\n");
    printf("i func %d\n", i);
    return answer;
}
```

rectFunction.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/7-3_functions/code/rectFunction.c)

An example of using functions to break down a nested loop into less complex parts

```c
#include <stdio.h>

//prints a row of size 'columns' of *
void printRow(int columns);
int main(void) {
    int i;
    int j;
    int rows;
    int cols;
    printf("Enter rows and cols\n");
    scanf("%d %d",&rows,&cols);
    i = 0;
    while(i < rows ){
       printRow(cols);
       printf("\n");
       i = i + 1;
    }
    printf("\nGoodbye\n");

    return 0;
}

//prints a row of size 'columns' of *
void printRow(int columns){
   int j;
   j = 0;
   while ( j < columns ){
      printf("*");
      j = j + 1;
   }
}
```

scope.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/7-3_functions/code/scope.c)

An example to illustrate variable scope and lifetime

```c
#include <stdio.h>

int n=5; // global variable, do not use those!
int bar(int n);  //prototype for bar
void foo(void);  //we added this prototype for foo

int main(void){
    //int i = j; //This will not compile
    int j = n;
    foo();
    printf("In main n is %d\n",n);
    return 0;
}

void foo(void){
    int n = 10;
    //int j = i; //This will not compile
    bar(n);
    printf("In foo n is %d\n",n);

}

int bar(int n){
    n = n + 5;
    printf("In bar n is %d\n",n);
    return ( n );
}
```

callByValue2.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/7-3_functions/code/callByValue2.c)

Simple example illustrating call by value

Try and trace through this program and understand what gets printed out and why.

```c
#include <stdio.h>

int f(int x);

int main(void) {
    int  x, y , z;
    x = 1;
    y = 2;
    z = f(y);

    printf("x=%d y=%d z=%d\n", x, y, z);
    return 0;
}


int f(int x){
    int y;
    x = x + 1;
    y = 3;
    return x * y;
}
```

scanfEx.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/7-3_functions/code/scanfEx.c)

Simple example of using scanf return value

```c
#include <stdio.h>

//Try these inputs
//1 2 3
//chicken
//1 2 chicken
//1 2 3chicken

//Try just typing Crl^d on a line on its own.

//Try typing
//2 3
//And then Ctrl ^d on a line of its own.


int main(void) {
    int itemsRead;
    int x,y,z;

    itemsRead = scanf("%d %d %d",&x,&y,&z);

    printf("I read in %d items: %d %d %d\n",itemsRead,x,y,z);

    return 0;
}
```

mainReturn.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/7-3_functions/code/mainReturn.c)

Read 5 numbers using scanf and print them

This version shows using return in main to end program execution early after an error.

Try entering in non-numeric data to see it end

```c
#include <stdio.h>
#include <stdlib.h>

#define N_NUMBERS 5

int main(void) {
    int x, i;

    printf("Enter %d numbers: ", N_NUMBERS);
    i = 0;
    while (i < N_NUMBERS) {
        if(scanf("%d", &x) != 1){
            printf("Invalid\n");
            //This is the same as
            //return -1;
            return EXIT_FAILURE;
        }
        printf("%d\n",x);
        i = i + 1;
    }

    printf("Goodbye\n");
    //This is the same as
    //return 0;

    return EXIT_SUCCESS;
}
```

lec_func4.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/7-3_functions/code/lec_func4.c)

```c
#include <stdio.h>

#define TRUE 1
#define FALSE 0

int biggerThanFive(int);

int main(void) {
    int number;
    scanf("%d", &number);
    printf("%d\n", biggerThanFive(number));
    return 0;
}

int biggerThanFive(int n) {
    int answer;
    if (n > 5) {
        answer = TRUE;
    } else {
        answer = FALSE;
    }
    printf("Done!\n");
    return answer;
}
```

lec_func.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/7-3_functions/code/lec_func.c)

```c
#include <stdio.h>

void plusOne(int);

int main(void) {
    int number;
    number = 4;
    printf("main number: %d\n", number);
    plusOne(number);
    printf("main number: %d\n", number);

    return 0;
}

void plusOne(int passedInNumber) {
    passedInNumber = passedInNumber + 1;
    printf("%d\n", passedInNumber);
}
```

returnVals.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/7-3_functions/code/returnVals.c)

Demonstration of a user defined function with a return value

```c
#include <stdio.h>

int f(int n);

int main(void) {
    int n = 1;

    //n = 6 + 10
    //n = 16
    n = f(n) + 10;
    printf("%d\n",n);

    return 0;
}

// input is 1
int f(int n){
    return (n + 5); //6
}
```

scanfLoop.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/7-3_functions/code/scanfLoop.c)

Simple example of using scanf return value to control a loop

This loop reads in until the user enters a non-integer value or until 10 ints have been read in

```c
#include <stdio.h>

//Try: Read in max of 10, report how many valid inputs it read.

#define INPUTS 10

int main(void) {
    int num;
    int counter;

    counter = 0;
    while(counter < INPUTS && scanf("%d",&num) == 1){
       printf("I read in %d\n",num);
       counter = counter + 1;
    }
    printf("\nI read in %d inputs\n",counter);
    return 0;
}
```

what1.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/7-3_functions/code/what1.c)

Exercise: Try to work out what this does without running it first then run to check

```c
#include <stdio.h>
#include <stdlib.h>

void f1(void);
void f2(void);
void f3(void);
void f4(void);
int main(void){
    printf("This ");
    f1();
    printf("how ");
    f4();
    printf("work!\n");
    return 0;
}

void f1(void){
    printf("is ");
    f2();
    printf("know ");
}

void f2(void){
     printf("to ");
     f3();
     printf("you ");
}

void f3(void){
    printf("ensure ");
}

void f4(void){
    printf("functions ");
}
```

printMessagesReturn.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/7-3_functions/code/printMessagesReturn.c)

Simple example of using functions

This uses return in a void function to terminate the function early under certain conditions

This could be rewritten using if statements

```c
#include <stdio.h>

void printManyMessages(int n);
void printMessages(void);

int main(void) {
    int n;
    printManyMessages(1);
    printf("Repeat this how many times? ");
    scanf("%d", &n);
    printManyMessages(n);
    return 0;
}

// This function demonstrates using a return statement
// to change the flow of control and exit the function
// before it reaches the end. See the alternative version
// below, restructured with the return statement
// Some consider this bad style and prefer the alternative version
// I think it is ok to use sparingly
void printManyMessages(int n) {
    if(n > 100){
        printf("No!\n");
        return;
    }
    while (n > 0) {
        printMessages();
        n = n - 1;
    }
    printf("Actually I am starting to feel a little C sick\n");
}

/*
Alternate version using if statements
void printManyMessages(int n) {
    if(n > 100){
        printf("No!\n");
    } else {
        while (n > 0) {
            printMessages();
            n = n - 1;
        }
        printf("Actually I am starting to feel a little C sickk\n");
    }
}
**/

void printMessages(void) {
    printf("C is good.\n");
    printf("C is great.\n");
    printf("We all love C.\n");
}
```

what.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/7-3_functions/code/what.c)

Exercise: What does this print?

Try to trace it through

```c
#include <stdio.h>
#include <stdlib.h>

int f1(int x);
int f2(int x, int y);

int main(int argc, char * argv[]){
    int num1 = 10;

    int num2 = f1(num1); //31
    printf("Answer is %d\n",num2);

    return 0;
}

//f1(10)
int f1(int x){
    int result = f2(x,x*2);
    result = result + 1;
    return result;

    //f2(10,20) = 30 + 1 = 31
    //return (f2(x,x*2) + 1);
}

//f2(10,20)
int f2(int x, int y){
    return x+y; //return 30
}
```

lec_func3.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/7-3_functions/code/lec_func3.c)

```c
#include <stdio.h>

int func(void);

int main(void) {
    printf("%d\n", func());
    return 0;
}

int func(void) {
    double x = 3.5;
    return x;
}
```

printMessages.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/7-3_functions/code/printMessages.c)

Simple example of using functions

```c
#include <stdio.h>

void printManyMessages(int n);
void printMessages(void);

int main(void) {
    int n;
    printManyMessages(1);
    printf("Repeat this how many times? ");
    scanf("%d", &n);
    printManyMessages(n);
    return 0;
}

void printManyMessages(int n) {
    while (n > 0) {
        printMessages();
        n = n - 1;
    }
    printf("Actually I am starting to feel a little C sick\n");
}

void printMessages(void) {
    printf("C is good.\n");
    printf("C is great.\n");
    printf("We all love C.\n");
}
```

functionTypes.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/7-3_functions/code/functionTypes.c)

Demonstration of user defined functions

```
#include <stdio.h>



void printStuff(void);
void printNumber(int n);
int getNumber(void);
int doCalculation(int n);


int main(void) {
    int result;
    printStuff();
    printNumber(5);
    result = getNumber();
    printf("result %d\n",result);

    result = doCalculation(99);
    printf("result %d\n",result);
    return 0;
}

void printStuff(void){
    printf("Stuff\n");

}

void printNumber(int n){
    int i = n;
    while(i > 0){
        printf("%d ",i);
        i = i - 1;
    }
    printf("Blast off!\n");

}

int getNumber(void){

   return 42;
}

int doCalculation(int n){
   return n+1;
}
```

mathEx.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/7-3_functions/code/mathEx.c)

Simple example of using library functions

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

//Note: If using gcc you may need the -lm flag
//gcc -Wall -Werror -O -o mathEx mathEx.c -lm

//Try abs with a double
int main(void) {
    int a = -199;
    int result = abs(a);

    double b = -199.123;
    //Using abs on a double will result in a compiler warning
    //(and a truncated answer)or with some compilers, an error.
    //There is one for double (floating point numbers) we use instead.
    double resultB = fabs(b);


    printf("Absolute Value of %d is %d\n",a,result);
    printf("The square root of 2 is %lf\n",sqrt(2));
    printf("Absolute Value of %lf is %lf\n",b,resultB);

    return 0;
}
```

feet2metresFunction.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/7-3_functions/code/feet2metresFunction.c)

Convert a measurement in feet to metres.

Modified feet2metres.c from C_basics lecture code to use a user defined function. Completed as a lecture exercise.

```
#include <stdio.h>

#define INCHES_IN_FOOT  12
#define CM_IN_INCH      2.54
#define CM_IN_METRE     100

//Converts input in feet to metres
double calcMetres (double feet);

int main(void) {
    double feet;
    double metres;

    printf("Enter number of feet: ");
    scanf("%lf", &feet);

    metres = calcMetres(feet);

    printf("%.2lf", feet);
    printf(" feet is ");
    printf("%.2lf", metres);
    printf(" metres\n");

    return 0;
}


//Converts input in feet to metres
double calcMetres (double feet){
   double m = feet * INCHES_IN_FOOT * CM_IN_INCH / CM_IN_METRE;
   return m;
}
```