C Programming - Midterm 1 Study Guide – Examples

Function Declarations

```
int sumOfTwo(int,int);
int nextFibonacci(int[]);
```

Function Definitions - Anywhere below the declarations

```
int sumOfTwo(int a, int b){
   return a+b;
}

//int state[2];
//state[0] = state[1] = 1;
int nextFibonacci(int state[]){
   int temp = state[1];
   state[1] = sumOfTwo(state[0], state[1]);
   state[0] = temp;
   return state[1];
}

int main(int argc, char* argv[]){
   // int argc, char* argv[]
   printf("The arguments provided are: ");
   int i;

        for(i=0; i < argc; i++){
                printf("%d)%s ", i, argv[i]);
         }
```

```
printf("\n");
```

Data Types and Sizes

```
char a = 'A';
char b[100]; strcpy(b, "bees?");
signed char c = 'C';
i = -1;
unsigned int j = 2;
long int k = 3000000000;
unsigned long x = 3000000001;
long y = x*k;
double z = (double)y;
double pi = 3.14159265359;
```

printf - similar to 'mad libs'

```
printf("The %s jumps over the %s\n", "quick brown fox", "lazy dog");

/* The first argument to printf is a format string, which acts as a template.
Like with mad libs, the format string indicates the type of thing to be inserted.
Variable types are very important with printf
*/

// printf - formatted printing on data types - mostly ignore
printf("%30c%15s%lu byte\n", a, "size = ", sizeof(a));
printf("%30s%15s%lu bytes\n", b, "size = ", sizeof(b));
printf("%30c%15s%lu byte\n", c, "size = ", sizeof(c));
printf("%30d%15s%lu bytes\n", i, "size = ", sizeof(i));
printf("%30d%15s%lu bytes\n", j, "size = ", sizeof(j));
```

```
    printf("%30ld%15s%lu bytes\n", k, "size = ", sizeof(k));
    printf("%30lu%15s%lu bytes\n", x, "size = ", sizeof(x));
    printf("%30ld%15s%lu bytes\n", y, "size = ", sizeof(y));
    printf("%30f%15s%lu bytes\n", z, "size = ", sizeof(z));
    printf("%30f%15s%lu bytes\n", pi, "size = ", sizeof(pi));

    if(0){
        printf("Let's try some formatted input.");
        printf("Please write the following exactly: 3.0*10**8 m/s\n");
        scanf("%s m/s\n", b);
        printf("%s was captured.\n", b); // Notice how only the sequence *before* m/s was captured

        fflush(stdin); // If there are input problems, you can try flushing the input buffer
        printf("Input an integer: ");
        scanf("%d\n", &i);
        printf("%d was captured.\n", i);
    }

    char buffer[]

    FILE * fpIn = fopen("character_map", "w");
```

LOOPS - For Loop

```
    int row, col;
    char l = 'A';
    for(row = 0; row < 3; row++){
        for(col = 0; col < 10; col++){
            printf("%4c", l);
            fprintf(fpIn, "%4c", l);
```

```
        l++;
    }
    printf("\n");
    fprintf(fpIn, "\n");
}
```

While Loop

```
int countdown = 10;
while(countdown){ // repetition condition can be an integer! 0 is false, anything else is true
    printf("%d\n", countdown--);
}
printf("Blast-off!\n");

double accel = 10; //m/s
double t = 0;
printf("%5s%10s\n", "time", "position");
printf("%5s%10s\n", "----", "--------");
do{
    printf("%5.2f%10.3f\n", t, .5*accel*t*t);
    t = t + .2;
}while(t < 5);
```

Do-While Loop

```
char buffer[BUFSIZ+1];
printf("Enter a sequence of values (-1 to stop): ");
int value;
int total = 0;
do{
```

```
        scanf("%d", &value);
        //value = atoi(fgets(buffer, BUFSIZ, stdin));
        if(total != -1)
            total = total + value;
    }while(value != -1); //continuation condition
    printf("Total %d\n", total);

    int state[2];
    state[0] = state[1] = 1;
    int fibonacci = 1;
    const int limit = 100;
    printf("Fibonacci Sequence:\n1\n1\n");
    fprintf(fpIn, "Fibonacci Sequence:\n1\n1\n");
    while(fibonacci < limit){
        printf("%d\n", fibonacci = nextFibonacci(state));
        fprintf(fpIn, "%d\n", fibonacci = nextFibonacci(state));
    }

    printf("Complete\n");

    fclose(fpIn);

    return 0;
}
```

Resources:

http://alvinalexander.com/programming/printf-format-cheat-sheet

https://www.tutorialspoint.com/cprogramming/c_arrays.htm

http://www.cprogramming.com/tutorial/c/lesson8.html

https://www.tutorialspoint.com/cprogramming/c_pointers.htm