Daniel Rao

CS-354

1. Functions in imperative programming language vs. functions in imperative.

In imperative programming language we may not get the same the same input parameters. We can have a math function that define an infinite set of inputs but in imperative programming language function our input is finite. Also, in imperative programming language we have functions that return nothing, but in math we can't have that. Also a math function always maps for the same value to the other regardless of how many times we repeat the process.

This may not be the same for imperative programming language. Here, is the example provided below:

```
int randfunc(int a) {
    return a * rand();
}
```

Other things include are that a math function requires an input value and must return same value that it evaluates to. However its not the same case for imperative programming language.

Apart from that a math function doesn't throw exception.

2 C program:

```
# include <stdio.h>


static void foo (int a, int b, int c) { }

static int argument (int n)
{
  printf ( Argument %d.\n", n);
        return n;
 }

 int main (void)
 {
     foo (argument (0), argument (1), argument (2));
    return 0;
  }
```

(3) The reason to why variable-length (argument list) are so seldom supported by high-level programming languages:

(a) It is not a good approach all times as it is vulnerable to errors and exceptions and needs excessive coding.

(b) The risk is that the value of one type is assumed when it belongs to another type. For example, a null pointer may be considered as an integer.