Week 1_2 class notes:

1.
```
int      main( ) {
printf("Hello world!\n");
return  0;
}
```

compile a C program: gcc hello.c

gcc -o hello hello.c

run a C program: ./hello

2. printf/scanf

```
int main() {
        int      i;  float  f;  char  c;
         i = 34;
        c = 'a';
        printf("%d\n",i);
        printf("%d\t%c\n",i,c);
}
```

Formatting instructions

- **Special characters**
  - **\n : newline**
  - **\t : tab**
  - **\b : backspace (which means cursor goes back a space to print next character)**
  - **\" : double quote**
  - **\\ : backslash**
- **Types of arguments**
  - **%d: integers**
  - **%f: floating-point numbers**
  - **%c: characters**
- **Formatting instructions**
  - **%6d: decimal integers at least 6 characters wide**
  - **%6f: floating point at least 6 characters wide**
  - **%6.2f: floating point at least 6 wide, 2 after the decimal point**

3. Data Types:

int

- integer: 16 bits or 32 bits (implementation dependent)

long

- integer: either 32 bits or 64 bits, depending on the architecture

long long

- integer: 64 bits

char

- a single byte

float

- single-precision floating point

double

- double-precision floating point

4. sizeof()

```
int main() {
        int i;
        printf("%d\n", sizeof(i));
}
```

5. Arrays in C: no boundary check

```
Int main() {
        int a[100];
        int i;
        for(i=0;i<=100;i++) a[i] = i;
}
```

6. assert()

```
#include <assert.h>
float   fact(int i) {
        int     k; float res;
        assert(i >= 0);
        for(res=0,k=1; k<=i; k++)
                res = res * k;
        return  res;
}
int main() {
        printf("%f\n", fact(5));
        return  0;
}
```

7. passing array as an parameter

```
int average(int a[], int size)
{  int     i; int     sum;
          for(i=0,sum=0; i<size; i++)
                    sum += a[i];
          return   sum/size;
}

void swap(int a[], int i, int j){
          int temp;
          temp = a[j];
          a[j] = a[i];
          a[i] = temp;
}

void selectsort(int array[], int length){
                    int i, j, min;
                    for (i = 0; i < length; ++i){
                    /* find the index of the smallest item from i onward */
                    min = i;
                    for (j = i; j < length; ++j)
                              if (array[j] < array[min])  min = j;
                    /* swap the smallest item with the i-th item */
                    swap(array, i, min);
                    }
          /* at the end of each iteration, the first i slots have the i  smallest items */
          }
```