# Activity 1 - Program Translation

Organize the following steps of program translation into the order that they occur. Linking, Compiling, Preprocessing, Assembling. Additionally, where does lexical, syntax, and semantic analysis take place? What are errors that can take place during each step of program translation?

# Activity 2 - Bitwise Operators

Your boss assigns you the task of reading in data from an accelerometer and pulling out the useful information. You look at the data sheet and see that each time you read from the accelerometer you get back an integer. Inside the integer is packed the status of the chip, the x acceleration, y acceleration, and z acceleration in that order. Write functions to pull out the information to match the code snippet below using the fact that the largest a byte can hold is 255.

```
int main() {
      uint32_t raw_value = read_acc(); // Assume this returns the value from the accelerometer
      uint8_t status = get_status(raw_value);
      uint8_t x_acc = get_x_acc(raw_value);
      uint8_t y_acc = get_y_acc(raw_value);
      uint8_t z_acc = get_z_acc(raw_value);

      printf("The original value was %#010X\n", raw_value);
      printf("The status was %#4X\n", status);
      printf("The x_acc was %#4X\n", x_acc);
      printf("The y_acc was %#4X\n", y_acc);
      printf("The z_acc was %#4X\n", z_acc);

      return 0;
}
```

# Activity 3 - System and Process Level IO

Fill in the table comparing system level IO to the process level IO that we use when writting code.

| | Our Program | OS |
|---|---|---|
| File Representation | | |
| Opening a File | | |
| Reading in byte oriented data | | |
| Return value when reading/writing byte oriented data | | |
| Uses buffering to reduce calls to lower level | | |
| Close a file | | |
| Manipulate offset in file | | |
| Flush buffer | | |

# Activity 4 - Unions

What is the output of the code snippet below.

```
#include <stdio.h>

union ex {
    int num;
    char c;
};

int main() {
    union ex my_union;
    my_union.num = 65;
    printf("The size of union ex is %lu\n", sizeof(union ex));
    printf("The char in my_union is %c\n", my_union.c);
    my_union.c = 'B';
    printf("The int in my_union is %d\n", my_union.num);
}
```

## Activity 5 - Pointers

Draw the memory diagram for the given program, is there an error? If there is, what is it? Otherwise what is the output.

```c
#include <stdio.h>

  int main() {
      int a = 10;
      int x = 20;
      int *b = &a;
      int **c = &b;
      **c = 20;
      b = &x;
      *c = 30;
      *b = 30;
      return 0;
  }
```

## Activity 6 - Enum Structs, Dynamic Memory

Implement the function make cat using the code below

```c
/** Create declaration of the enum below **/









typedef struct cat_s {
    char *name;
    int age;
    enum Color color;
    void (*meow)();
} Cat;

void cat_speak() {
    printf("Meow\n");
}

Cat *make_cat(char *name, int age, enum Color color) {









}
```

## Activity 7 - Processes

Create a process that outputs the current working directory. You can use the command pwd to get the current directory.