Activity 1 - Grab Bag

Take one card from the pile. You will have 5 minutes to "research" the topic by going through your notes before presenting to the rest of the group everything you found on the topic. Don't forget to use the topic list for Exam 2 and the class lecture slides.

Activity 2 - Structs and Function Pointers

Create a struct "student" that has a name, age, gpa, and a function pointer that can compare if two students are equal. Two students are equal if they have the same name, age, and gpa. Then make the "create student" function that uses dynamic memory. The following code snippet shows the struct "student" in use.

```
int main() {
    struct student *student_1 = create_student("Bob", 19, 3.5);
    struct student *student_2 = create_student("Sam", 20, 4.0);
    struct student *student_3 = create_student("Bob", 19, 3.5);

    if(student_1->equal(student_1, student_2))
        printf("Student 1 and Student 2 are equal\n");
    if(student_1->equal(student_1, student_3))
        printf("Student 1 and Student 3 are equal\n");

    return 0;
}
```

Activity 3 - Determing Size of Struct

Determine the size of each of the following structs. When you feel comfortable with one of the answers write it on the board including the total number of bytes are padding.

```
struct person_s {
    char favorite_letter;
    char *name;
    long id;
};

struct dog_s {
    char *name;
    void (*bark)();
};

struct house_s {
    int number;
    char street_name[20];
    char owner_name[10];
};

int main() {
    printf("The size of a person is %lu\n", sizeof(struct person_s));
    printf("The size of a dog is %lu\n", sizeof(struct dog_s));
    printf("The size of a house is %lu\n", sizeof(struct house_s));
    return 0;
}
```

Activity 4 - Compilation Steps

For each of the following steps, write an invalid C statement that would be caught by that analyzer.

1. Lexical Analyzer

2. Syntax Analyzer

3. Semantic Analyzer

Activity 5 - C Preprocessor Practice

Convert the following code to only use C preprocessor macros. Remember that you can have conditionals and macros that take in parameters using the C preprocessor. You can leave main as a normal function and the variable. All other functions should be converted to preprocessor macros.

```c
int max(int a, int b) {
    return a > b ? a : b;
}

int is_valid(int a) {
    return a < 10;
}

int main() {
    int a = 50;
    int b = 5;

    printf("The biggest number is %d\n", max(a, b));

    if(is_valid(b)) {
        printf("The value of b is less than 10;
    }

    return 0;
}
```

Activity 6 - ADT Practice

Create an ADT for an array that can be used in the following way. What makes the array an ADT? What are some methods to improve the ADT? The ADT only has to hold ints.

```c
#include "array.h"
#include <stdio.h>

int main() {
    // Create an array that has size of 5
    Array my_array = create_array(5);
    // Set index 0 to 7
    set_element(my_array, 0, 7);
    printf("The element at index 0 is %d\n", get_element(my_array, 0));
    return 0;
}
```

Activity 7 - IO Practice

Create a program that is capable to echoing the user input from stdin to stdout. Look over the lecture notes and the topic list to see the options for IO that are covered in the exam.