Solutions to Functional Parameters Exercises

1. Write a function that takes a function and an integer as arguments, and returns the result of applying the function to the integer. The functional parameter should take one integer as its argument, and return an integer.

```
int
apply(int (*function)(int), int num)
{
  return(function(num));
}
```

2. Write a function that takes a function and two integers as arguments, and returns the result of applying the function to the larger of the two integers. The functional parameter should take one integer as its argument, and return an integer.

```
int
apply_to_max(int (*function)(int), int num1, int num2)
{
  if (num1 > num2)
    return(function(num1));
  else return(function(num2));
}
```

3. Write a function that takes two functions and an integer as arguments, and returns the maximum of the values returned by the two functions when applied to the integer. The functional parameters should each take one integer as argument, and return an integer.

```
typedef int (*int_int_fn)(int);

int
max_of_apply(int_int_fn function1, int_int_fn function2, int num)
{
  if (function1(num) > function2(num))
    return(function1(num));
  else return(function2(num));
}
```

Notice that this implementation of `max_of_apply` causes either `function1` or `function2` to be called twice. Although this is inefficient, it is slightly clearer than its more efficient siblings. I've used `typedef` here arbitrarily; it could have been used for the preceding functions as well.

4. Write a program that calls each of the above functions on sample data.

```
int
funny1(int num)
{
  return (num + 17);
}

int
funny2(int num)
{
  return(42 - num);
}

main()
{
  printf("apply(funny1, 12) = %d\n",
         apply(funny1, 12));
  printf("apply_to_max(funny1, 12, 88) = %d\n",
         apply_to_max(funny1, 12, 88));
  printf("max_of_apply(funny1, funny2, 12) = %d\n",
         max_of_apply(funny1, funny2, 12));
}
```

This program causes the following to be printed:

```
apply(funny1, 12) = 29
apply_to_max(funny1, 12, 88) = 105
max_of_apply(funny1, funny2, 12) = 30
```