1. What features of the C language make a buffer overflow attack possible?

   C compilers do not check for reading or writing beyond the bounds of an array.

2. Answer the following questions concerning how a program is stored in memory during its execution.

   a. Which segment of memory has contents that remain unchanged during program execution?

      The Text segment

   b. Does the programmer have complete control over how the stack is organized?

      No. While the programmer has control over the order of declared variables, ultimately the compiler chooses the size, position and organization of the stack.

   c. What is the relationship between the order in which variables appear in a function and the order in which these same variables are stored in a function’s stack frame?

      Variables are stacked according to the order in which they are declared, with the first declared variable stored near the bottom of the stack (at higher addresses), and subsequently declared variables stored up the stack (at lower addresses).

   d. What important registers are used to define the boundaries of a stack frame?

      ebp (base pointer) and esp (stack pointer)

   e. Suppose main calls a function named fun. After all the commands of fun have been executed, how does the program know how to continue at the exact location in main where it left off?

      While calling fun function, the next instruction in main function will be saved on the stack as the return address, so that when fun function is over, execution will resume at the saved return address.

   f. Is a source code file permitted to have more than one function?

      Yes. There is no limit to the number of functions a program may have.

   g. If your answer to (f) was “no,” explain why that is the case. If your answer to (f) was “yes,” explain how the operating system knows where to begin executing your program if the source code file contains multiple functions.

      Regardless of how many functions are declared, the operating system always begins executing C programs at the main() function.
3. Given the following code snippet:

```c
char first_name[6] = “Alice”;  
strcpy(first_name, “Alexander”);
```

(a) Will the compiler state that there is an error?

No. C does not check for array bounds.

(b) What potentially dangerous situation occurs because of the snippet above?

It results in a buffer overflow due to copying a longer string over a shorter one.

(c) What is the minimum size necessary for the array `first_name` to prevent this?

10 bytes

(d) There are at least two ways to change the above code to prevent the above error from happening. Describe one.

1. Declare the size of `first_name` to be at least 10 bytes.
2. Shorten Alexander to Alex.
3. Recode the program in another language, such as Java, C# or Python.
4. (not in the text) Replace `strcpy` with the safer `strncpy()`, which limits (truncates) the source string to n bytes.

4. When the greetings function is called in main from the following code sample, the stack pictured in the figure that follows is created.

```c
#include<stdio.h>
void greetings()
{
    int name_len = 15;
    char name[name_len];
    int year = 2014;

    printf( “Enter your name:” );
    scanf( “&s”, name );
    printf(“Hello: %s! The current year is %d.
”, name, year);
}

int main()
{
    greetings();
}
```
(a) Assuming there is no padding when the stack frame is created, how many characters must the user enter to overwrite only the first byte of the return address?

<table>
<thead>
<tr>
<th>Stack</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>year</td>
<td>15 bytes</td>
</tr>
<tr>
<td>name</td>
<td></td>
</tr>
<tr>
<td>name_len</td>
<td>4 bytes</td>
</tr>
<tr>
<td>previous ebp</td>
<td>4 bytes</td>
</tr>
<tr>
<td>return address</td>
<td></td>
</tr>
</tbody>
</table>

User must enter exactly 23 characters. The NULL will go into the 24th byte, which is the first byte of the return address.

(b) Is it possible to change the value of year by performing a buffer overflow attack? Why or why not?

No, because year is the last variable declared, which means it will be at the top of the stack, and therefore not vulnerable to buffer overflow attack.