Study Questions: Set No. 8
Introduction to C
Thursday November 14, 2013

Covering:

Chapter 13

1. The following function calls supposedly write a single new-line character, but some are incorrect. Identify which calls don't work and explain why.

```
(a) printf("%c", '\n');
(b) printf("%c", "\n");
(c) printf("%s", '\n');
(d) printf("%s", "\n");
(e) printf('\n');
(f) printf("\n");
(g) putchar('\n');
(h) putchar("\n");
(i) puts('\n');
(j) puts("\n");
(k) puts("");
```

2. Suppose that `p` has been declared as follows:

```
char *p = "abc";
```

Which of the following function calls are legal?
Show the output produced by each legal call, and explain why the others are illegal.

```
(a) putchar(p);
(b) putchar(*p);
(c) puts(p);
(d) puts(*p);
```

3. What is the output of the following C statements?

```
printf("*%20s*\n*%-20s*\n*%20s*\n*%-20s*\n",
       "ABCDEFGH", "ABCDEFGH", "ABCDEFGH", "ABCDEFGH");
```

4. What is the output of the following C statements?

```
printf("*%.5s*\n*%-.5s*\n*%.5s*\n*%-.5s*\n",
       "ABCDEFGH", "ABCDEFGH", "ABCDEFGH", "ABCDEFGH");
```

5. What is the output of the following C statements?

```
printf("*%20.5s*\n*%-20.5s*\n*%20.5s*\n*%-20.5s*\n",
       "ABCDEFGH", "ABCDEFGH", "ABCDEFGH", "ABCDEFGH");
```

6. Suppose that we call `scanf` as follows:
```
scanf ("%d%s%d", &i, s, &j);
```
If the user enters `12abc34 56def78`, what will be the values of `i`, `s`, and `j` after the call?
Assume that `i` and `j` are `int` variables and `s` is an array of characters.

7. Write a function named `capitalize` that capitalizes all letters in its argument. The argument will be a null-terminated string containing arbitrary characters, not just letters. Use array subscripting to access the characters in the string. *Hint:* Use the `toupper` function to convert each character to upper-case.

8. Rewrite the `capitalize` function, this time using pointer arithmetic to access the characters in the string.

9. Write a function named `censor` that modifies a string by replacing every occurrence of `foo` by `xxx`.
For example, the string `"food fool"` would become `"xxxd xxxl"`.
Make the function as short as possible without sacrificing clarity.

10. What will be the value of the string `s1` after the following statements have been executed?
```
strcpy(sl, "computer");
strcpy(s2, "science") ;

if (strcmp(sl, s2) < 0)
  strcat (s1, s2);
else
  strcat (s2, sl);
s1[strlen(s1)-6] = '\0';
```

11. Suppose that `str` is an array of characters. Which one of the following statements is not equivalent to the other three?
```
(a) *str = 0;
(b) str[0] = '\0'
(c) strcpy(str, "");
(d) strcat(str, "");
```

12. What will be the value of the string `str` after the following statements have been executed?
```
strcpy(str, "tire-bouchon");
strcpy (&str[4], "d-or-wi");
strcat(str, "red?");
```

13. The following function supposedly creates an identical copy of a string. What's wrong with the function?
```
char *duplicate(const char *p)
{ char *q;

  strcpy (q, p);
  return q;
}
```

14. Write the following function:
```
void get_extension(const char *filename, char *extension);
```
`file_name` points to a string containing a file name. The function should store the extension on the file name in the string pointed to by `extension`. For example, if the file name `is "memo.txt"`, the function will store `"txt"` in the string pointed to by `extension.` If the file name doesn't have an extension, the function should store an empty string (a single null character) in the string pointed to by `extension.` Keep the function as simple as possible by having it use the `strlen` and `strcpy` functions.

15. Write the following function:
`bool test_extension(const char *file_name, const char *extension);`
`filename` points to a string containing a file name. The function should return `true` if the file's extension matches the string pointed to by `extension,` ignoring the case of letters. For example, the call `test_extension ("memo.txt", "TXT")` would return `true.` Incorporate the "search for the end of a string" idiom into your function. *Hint:* Use the `toupper` function to convert characters to upper-case before comparing them.

16. Let `f` be the following function:

```
int f(char *s, char *t)
{
  char *p1, *p2;

  for(p1 = s; *p1; pl++)
  {
    for(p2 = t; *p2; p2++)
      if (*pl == *p2) break;
    if (*p2 == '\0') break;
  }
  return pl - s;
}
```
(a) What is the value of `f("abed", "babe")`?
(b) What is the value of `f("abed", "bcd")`?
(c) In general, what value does `f` return when passed two strings `s` and `t`?

17. What does the following program print?
```
#include <stdio.h>
int main(void)
{
  char s[] = "Hsjodi", *p;

  for(p = s; *p; p++)
    --*p;
  puts(s);
  return 0;
}
```

18. Write a program that finds the "smallest" and "largest" in a series of words. After the user enters the words, the program will determine which words would come first and last if the words were listed in dictionary order. The program must stop accepting input when the user enters a four-letter word. Assume that no word is more than 20 letters long. An interactive session with the program might look like this:
```
Enter word: dog
Enter word: zebra
Enter word: rabbit
Enter word: catfish
Enter word: walrus
Enter word: cat
Enter word: fish
Smallest word: cat
Largest word: zebra
```
Hint: Use two strings named `sma1lest_word` and `1argest_word` to keep track of the "smallest" and ,"largest" words entered so far. Each time the user enters a new word, use `strcmp` to compare it with `sma1lest_word;` if the new word is "smaller," use `strcpy` to save it in `smallest_word.` Do a similar comparison with `largest_word.` Use `str1en` to determine when the user has entered a four-letter word.

19. Write a program named `reverse.c` that echoes its command-line arguments in reverse order. Running the program by typing

```
reverse void and null
```

should produce the following output:

```
null and void
```

20. Write a program named `sum.c` that adds up its command-line arguments, which are assumed to be integers. Running the program by typing

```
sum 8 24 62
```

should produce the following output:

```
Total: 94
```

Hint: Use the `atoi` function to convert each command-line argument from string form to integer form.

21. What string is displayed by the following code fragment?

```
char tmp1[40], tmp2[40], colors[] = "Red Green Blue";
strncpy(tmp1, &colors[4], 5);
tmp1[5] = '\0';
strcat(tmp1, " ");
strncpy(tmp2, colors, 3);
tmp2[4] = '\0';
printf("%s\n", strcat(tmp1, tmp2));
```

22. What string is displayed by the following code fragment?

```
char *s;
char p[] = "I have too much work";
s = &p[3];
printf("%s\n", &s[4]);
```

23. What string is displayed by the following code fragment?

```
char text[6] = "ABCDEF";
for (char *ptr = &text[1]; *ptr != ‘\0’; ++ptr)
{
   printf("%s\n", ptr);
}
```

24. What string is displayed by the following code fragment?

```
char s5[5], s10[10], s20[20], a_day[10] = "Sunday", another[] = "Saturday";

strncpy(s5, another, 4);
s5[4] = '\0';

strcpy(s10, &a_day[3]);

strcpy(s20, a_day);
strcat(s20, another);

printf("%d\n", strlen(a_day));
printf("%d\n", strlen(another));

printf("%s\n", s5);
printf("%s\n", s10);
printf("%s\n", s20);

printf("%s\n", &s5[0]);
printf("%s\n", &s10[0]);
printf("%s\n", &s20[0]);
```

25. What string is displayed by the following code fragment?

```
char text[18] = "Rust never sleeps", * ptr;

for (ptr = &text[16]; ptr != text; ptr -= 2)
{
 printf("%s\n", ptr);
}
```

26. What string is displayed by the following code fragment?

```
char text[18] = "Rust never sleeps", * ptr;

for (ptr = &text[16]; ptr != text; ptr -= 2)
{
 printf("%c\n", *ptr);
}
```

27. Give the output of the following program fragment

```
int main()
{ char text[] = "ABCDEFGHIJK";
  char *ptr = &text[1];

  while( *ptr++ != '\0')
    printf("%s\n", ptr++);
}
```

28. Give the output of the following program fragment

```
int main()
{ char *s[5] = {"ABCDEF","GHIJ", "KLMNOPQ", "RSTUV", "WXYZ"};

  printf("%s\n", s[2]);
  printf("%s\n", &s[3][2]);
  printf("%s\n", &s[2][3]);
}
```

29. Give the output of the following program fragment

```
int main()
{ char *s[5] = {"ABCDEF","GHIJ", "KLMNOPQ", "RSTUV", "WXYZ"};

  printf("%s\n", &(*s)[2]);
  printf("%s\n", *(s+2));
  printf("%s\n", (*s+2));
}
```

30. Give the output of the following program fragment

```
int main()
{ char *s[5] = {"ABCDEF","GHIJ", "KLMNOPQ", "RSTUV", "WXYZ"};
  char **p[5] = { &s[4], &s[3], &s[0], &s[2], &s[1] };

  printf("%s\n", &(**p[3]));
  printf("%s\n", *p[3]);
  printf("%s\n", &((**p)[3]));
}
```

31. Choose the correct answer for the outcome of the following program fragment

```
int main()
{ char *text = "ABCD", *p = "JKLM";

  p = text;
  p[2] = 'Z';
  printf("%s\n", text);
}
```

a) ABCD b) ABZD c) JKLM d) JKZM e) Won't Run

32. Choose the correct answer for the outcome of the following program fragment

```
int main()
{ char text[] = "ABCD", *p = "JKLM";

  p = text;
  p[2] = 'Z';
  printf("%s\n", text);
}
```

a) ABCD b) ABZD c) JKLM d) JKZM e) Won't Run

33. Choose the correct answer for the outcome of the following program fragment

```
int main()
{ char *text = "ABCD", *p = "JKLM";

  text = p;
  p[2] = 'Z';
  printf("%s\n", text);
}
```

a) ABCD b) ABZD c) JKLM d) JKZM e) Won't Run

34. Choose the correct answer for the outcome of the following program fragment

```
int main()
{ char text[] = "ABCD", *p = "JKLM";

  text = p;
  p[2] = 'Z';
  printf("%s\n", text);
}
```

a) ABCD b) ABZD c) JKLM d) JKZM e) Won't Run

35. Give the output of the following program fragment

```
int main()
{ char text[] = "ABCDEFGH";
  char*p = text + 2;
  int sum  = 0, i = 0, count = 0;

  for (i=0; i < 4; i++)
  { sum += p[i];
    count++;
  }
  printf("%c\n", sum/count);
}
```

36. Give the output of the following program fragment

```
int main()
{
  char text[] = "ABCDEFGH";
  char *p = text + 2;
  int sum  = 0, i = 0, count = 0;

  for (i=0; i < 4; i++)
  {  sum += *(p + i);
     count++;
  }
  printf("%c\n", sum/count);
}
```

37. Give the output of the following program fragment

```
int main()
{
  char text[] = "02468";
  char *p = text + 2;
  int sum  = 0, i = 0, count = 0;

  for (i=1; i < 3; i++)
  {  sum += *(p + i);
     count++;
  }
  printf("%c\n", sum/count);
}
```

38. Give the output of the following program fragment

```
int main()
{
  char text[] = "02468";
  char *p = text + 2;
  int sum  = 0, i = 0, count = 0;

  for (i=1; i < 3; i++)
  {  sum += *p + i;
     count++;
  }
  printf("%c\n", sum/count);
}
```

39. Write a program that takes in an undefined number of words from the user and then merges all of the words letter by letter. For example if the program is called merge we could type on the command line:
`merge Aaaaa Bbb Cccc`
and the resulting output would be: `ABCabcabcaca`