Study Questions: Set No. 6
Introduction to C
Tuesday October 29, 2013

Covering:

Chapter 8, 9 and Chapter 10

1. Write a declaration of an array named `weekend` containing seven `bool` values. Include an initializer that makes the first and last values `true`; all other values should be `false`.

2. Identify the error in the following C statement:
```
int x[8], i;

for(i = 0; i<=8; ++i)
   x[i] = i;
```
   Will the error be detected? If so, when?

3. What is the output of the `printf` statement in the following portion of a C program?
```
{
     int x[10]={0,5,10,15};
     int  k=1,  sum;

     for(sum=0;k<4;k++)
     {
       x[k]=  k/2;
       sum  +=  x[k];
       sum  +=  x[k+1];

       printf("%d\n", sum);
     }
}
```

4. Write a program segment to display the sum of the values in each row of a 5×3 type `double` array named `table`. How many row sums will be displayed? How many elements are included in each sum?

5. Re-answer the previous question for the column sums.

6. The Febonacci number are 0, 1, 1, 2, 3, 5, 8, 13, … where each number is the sum of the two preceding numbers. Write a program fragment that declares an array named `fib_numbers` of length 40 and fills the array with the first 40 Febonacci numbers. Hint: Fill in the first two numbers individually, then use a loop to compute the remaining numbers.

7. Write a C program segment to display the index of the smallest and the largest numbers in an array *x* of 20 integers. Assume array *x* already have values assigned to each element.

8. Write a declaration for a two-dimensional array named `temperature_readings` that stores one month of hourly temperature readings. For simplicity, assume that a month has 30 days. The rows of the array should represent days of the month; the columns should represent hours of the day.

9. Using the array of the above exercise, write a program fragment that computes the average temperature for a month (averaged over all days of the month and all hours of the day).

10. Write a declaration for an 8×8 `char` array named `chess_board`. Include an initializer that puts the following data into the array (one character per array element):

```
r   n   b   q   k   b   n   r
P   p   p   p   p   p   p   p
.   .   .   .   .   .   .   .
.   .   .   .   .   .   .   .
.   .   .   .   .   .   .   .
.   .   .   .   .   .   .   .
P   p   p   p   p   p   p   p
R   N   B   Q   K   B   N   R
```

11. Write a program fragment that declares an 8 × 8 `char` array named `checker_board` and then uses a loop to store the following data into the array (one character per array element):

```
B   R   B   R   B   R   B   R
R   B   R   B   R   B   R   B
B   R   B   R   B   R   B   R
R   B   R   B   R   B   R   B
B   R   B   R   B   R   B   R
R   B   R   B   R   B   R   B
B   R   B   R   B   R   B   R
R   B   R   B   R   B   R   B
```

Hint: The element in row `i`, column `j`, should be the letter B if `i + j` is an even number.

12. Write a program that reads a 5×5 array of integers and then prints the row sums and the column sums:

```
Enter row 1: 8 3 9 0 10
Enter row 2: 3 5 17 1 1
Enter row 3: 2 8 6 23 1
Enter row 4: 15 7 3 2 9
Enter row 5: 6 14 2 6 0

Row totals: 30 27 40 36 28
Column totals: 34 37 37 32 21
```

13. Write a program that prints a one-month calendar. The user specified the number of days in the month and the day of the week on which the month begins:

```
Enter number of days in month: 31
Enter starting day of the week (1= Sunday, 2=Monday,... 7=Saturday): 3
             1    2    3    4    5
 6      7    8    9   10   11   12
13     14   15   16   17   18   19
20     21   22   23   24   25   26
27     28   29   30   31
```

Hint: This program is not as hard as it looks. The most important part is a `for` statement that uses a variable `i` to count from `1` to `n`, where `n` is the number of days in the month, printing each value of `i`. Inside the loop, an `if` statement tests whether `i` is the last day in a week; if so, it prints a new-line character.

14. One of the oldest known encryption techniques is the Caesar cipher, attributed to Julius Caesar. It involves replacing each letter in a message with another letter that is a fixed number of positions later in the alphabet. If the replacement would go past the letter Z, the cipher "wraps around" to the beginning of the alphabet. For example, if each letter is replaced by the letter two positions after it, then Y would be replaced by A, and Z would be replaced by B. Write a program that encrypts a message using a Caesar cipher. The user will enter the message to be encrypted and the shift amount (the number of positions by which letters should be shifted):

```
Enter message to be encrypted: Go ahead, make my day.
Enter shift amount (1-25): 3
Encrypted message: Jr dkhdg, pdnh pb gdb.
```

Notice that the program can decrypt a message if the user enters 26 minus the original key:

```
Enter message to be encrypted: Jr dkhdg, pdnh pb gdb.
Enter shift amount (1-25):23
Encrypted message: Go ahead, make my day.
```

You may assume that the message does not exceed 80 characters. Characters other than letters should be left unchanged. Lower-case letters remain lower-case when encrypted and upper-case letters remain upper-case. Hint: To handle the wrap-around problem, use the expression `((ch -'a') + n) % 26 + 'A'` to calculate the encrypted version of an upper-case letter, where `ch` stores the letter and `n` stores the shift amount. (You'll need a similar expression for lower-case letters.)

15. Write a program that generates a "*random walk*" across a 10 x 10 array. The array will contain characters (all '.' initially). The program must randomly "walk" from element to element, always going up, down, left, or right by one element. The elements visited by the program will be labeled with the letters A through Z, in the order visited. Here is an example of the desired output:

```
A . . . . . . . . .
B C D . . . . . . .
. F E . . . . . . .
H G . . . . . . . .
I . . . . . . . . .
J . . . . . . . Z .
K . . R S T U V Y .
L M P Q . . . W X .
. N O . . . . . . .
. . . . . . . . . .
```

Hint: Use the `srand` and `rand` functions to generate random numbers. After generating a number, look at its remainder when divided by 4. There are four possible values for the remainder, 0, 1, 2, and 3, indicating the direction of the next move. Before performing a move, check that
(a) it will not go outside the array, and
(b) it does not take us to an element that already has a letter assigned.
If either condition is violated, try moving in any of the other directions. If all four directions are blocked, the program must terminate. Here is an example of premature termination:

```
A B G H I . . . . .
. C F . J K . . . .
. D E . M L . . . .
. . . . N O . . . .
. . W X Y P Q . . .
. . V U T S R . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
```

Y is blocked on all four sides, so there is no place to put Z.

16. Write a program that reverses the words in a sentence:

```
Enter a sentence: you can cage a swallow can't you?
Reversal of sentence: you can't swallow a cage can you?
```

Hint: Use a loop to read the characters one by one and store them in a one-dimensional `Char` array. Have the loop stop at a period, question mark, or exclamation point (the "terminating character"), which is saved in a separate char variable. Then use a second loop to search backward through the array for the beginning of the last word. Print the last word, and then search backward for the next-to-last word. Repeat until the beginning of the array is reached. Finally, print the terminating character.

17. Write a program that tests whether two words are anagrams (permutations of the same letters):

```
Enter first word: smartest
Enter second word: mattress
The words are anagrams.

Enter first word: dumbest
Enter second word: stumble
The words are not anagrams.
```

Write a loop that reads the first word, character by character, using an array of 26 integers to keep track of how many times each letter has been seen. (For example, after the word *smartest* has been read, the array should contain the values 10001000000010000122000000, reflecting the fact that *smartest* contains one *a*, one *e*, one *m*, one *r*, two *s*'s and two *t*'s.) Use another loop to read the second word, except this time decrementing the corresponding array element as each letter is read. Both loops should ignore any characters that aren't letters, and both should treat upper-case letters in the same way as lower-case letters. After the second word has been read, use a third loop to check whether all the elements in the array are zero. If so, the words are anagrams. *Hint*: You may wish to use functions from `<ctype.h>` such as `isalpha` and `tolower`.

18. The following function attempts to compute the area of a triangle. Yet, it contains two errors. Locate the errors and show how to fix them.

```
double triangle_area (double base, height)
double product;
{
  product = base * height;
  return product / 2;
}
```

19. What is wrong with the following C function?

```
int square(int x);
{
  return x*x;
}
```

20. What is the output of calling `show(4)`?

```
int show(int x)
{
   printf("%d %d\n", x, x*x);
   return x*x;
   printf("%d %d\n", x, x*x*x);
   return x*x*x;
}
```

21. What does the following C function do?
```
int eq3(int a, int b, int c)
{
  if((a == b) && (a == c))
    return 1;
  else
    return 0;
 }
```

22. What is the output of the following C program?
```
#include <stdio.h>

int f(int x)
{
  return x + 2;
}

int main(void)
{  int x = 5;
   printf("%d %d\n", f(x+2), f(f(x+2)));
   return 0;
}
```

23. What is the output of the following C program?
```
#include <stdio.h>

int confusion(int x, int y)
{  x = 2*x + y;
   return x;
}

int main(void)
{  int x = 2, y = 5;
   y = confusion(y, x);
   x = confusion(y, x);
   printf("%d %d\n", x, y);
   return 0;
}
```

24. What is the output of the following C program?
```
#include <stdio.h>
void swap(int a, int b);
int main (void)
{
  int i=1, j =2;
  swap (i , j) ;
  printf("i = %d,  j = %d n", i, j);
 return 0;
}

void(swap(int a, int b)
{
  int temp = a;
  a = b
  b = temp;
}
```

25. Write a C function that takes two integers as arguments and returns the value of the larger one.

26. Write a C function that takes three integers as arguments and returns the value of the largest one.

27. Write a C function that takes a real number as an argument and returns the absolute value of that number.

28. Write a C function that takes a positive integer *n* as an argument and returns the largest power of two smaller than or equal to *n*.

29. Write a C function that takes a positive integer *n* as an argument and returns 1 if *n* is prime, and 0 otherwise.

30. Write a function that takes a positive integer as input and returns the leading (the most significant) digit in its decimal representation. For example, the leading digit of 234567 is 2.

31. Write a function `digit(n, k)` that returns the k^{th} digit (from the right) in `n` (a positive integer). For example, `digit(829, 1)` returns `9`, `digit(829, 2)` returns `2`, and `digit(829, 3)` returns `8`. If `k` is greater than the number of digits in `n`, have the function return `0`.

32. Write a function `check(x, y, n)` that returns `1` if both `x` and `y` fall between `0` and `n-1`, inclusive. The function should return `0` otherwise. Assume that `x, y`, and `n` are all of type `int`.

33. Write a function `day_of_year(month, day, year)` that returns the day of the year (an integer between 1 and 366) specified by the three arguments.

34. Write a function to determine the number of prime numbers below *n*.

35. Write a function `num_digits(n)` that returns the number of digits in `n` (a positive integer). Hint: To determine the number of digits in a number `n`, divide it by 10 repeatedly. When `n` reaches `0`, the number of divisions indicates how many digits `n` originally had.

36. Write a function to raise a floating point number to an integer power. The prototype of the function should be:
`float raise_to_power(float f, int power);`
For example, `raise_to_power(2, 3)` returns `8`, and `raise_to_power(9, 2)` returns `81`.

37. Write a function that outputs a right isosceles triangle of height and width *n*, so the output for $n = 6$ should be

```
*
**
***
****
*****
******
```

38. Write a function that outputs a sideways triangle of height $2 \times n-1$ and width *n*, so the output for $n = 4$ should be:

```
*
**
***
****
***
**
*
```

39. Write a function that outputs a right-side-up triangle of height *n* and width *2×n-1*; the output for *n = 6* would be:

```
     *
    ***
   *****
  *******
 *********
***********
```

40. Write the following function, which evaluates a chess position:
`int evaluate_position(char board [8][8]);`
`board` represents a configuration of pieces on a chessboard, where the letters `K, Q, R, B, N,` and `P` represent *white* pieces. and the letters `k, q, r, b, n,` and `p` represent *black* pieces. `evaluate_position` should sum the values of the *white* pieces `(Q = 9, R = 5, B = 3, N = 3,` and `P = 1`). It should also sum the values of the *black* pieces (done in a similar way). The function will return the difference between the two numbers. This value will be positive if *white* has an advantage in material and negative if *black* has an advantage.

41. Write functions that return the following values. Assume that `a` and `n` are parameters, where `a` is an array of `int` values and `n` is the length of the array.
(a) The largest element in `a`
(b) The average of all elements in `a`
(c) The number of positive elements in `a`

42. Write the following function:
`double inner_product (double a[], double b[], int n);`
The function should return `a[0] × b[0] + a[1] × b[1] + ... + a[n-1] × b[n-1]`.

43. Suppose that the function `f` has the following definition:
`int f (int a, int b) {...}`
Which of the following statements are legal? (Assume that `i` has type `int` and `x` has type `double`)

```
i = f(83, 12);
x = f(83, 12);
i = f(3.15, 9.28);
x = f(3.15, 9.28);
f(83, 12);
```

44. Which of the following would be valid prototypes for a function that returns nothing and has one double parameter?

```
void f(double x);
void f(double);
void f(x);
f(double x);
```

45. The following function is supposed to return `true` if any element of the array `a` has the value 0 and `false` if all elements are nonzero. Sadly, it contains an error. Find the error and show how to fix it:

```
bool has_zero (int a[] , int n)
{
  int i;

  for (i = 0; i < n; i++)
  if (a[i] == 0)
    return true;
  else
    return false;
}
```

46. What are the output of the following program fragment?
```
void silly(int x)
{
  int y;
  y = x + 2;
  x *= 2;
}

int main(void)
{
  int x, y;

  x = 10; y = 11;
  silly(x);
  silly(y);
  printf("%d %d\n", x, y);
  return 0;
}
```

47. Trace the execution of the following function by hand. Then write a program that calls the function, passing it a number entered by the user. What does the function do?
```
void pb(int n)
{
  if (n != 0)
  {
    pb(n / 2);
    putchar('0' + n%2);
  }
}
```

48. The following function finds the median of three numbers. Rewrite the function so that it has just one `return` statement.
```
double median(double x, double y, double z)
{
if(x <= y)
if(y <= z) return y;
else if(x <= z) return z;
else return x;
if (z <= y) return y;
if (x <= z) return x;
return z;
}
```

49. Write the following function:
`float compute GPA(char grades[], int n);`
The `grades` array will contain letter grades (`A`, `B`, `C`, `D`, or `F`, either upper-case or lower-case), and `n` is the length of the array. The function should return the average of the grades (assume that `A=4`, `B=3`, `C=2`, `D=1`, and `F=0`).

50. Write a program that asks the user to enter a series of integers (to be stored in an array), then sorts the integers by calling the function `selection_sort`. When given an array with `n` elements, `selection_sort` must do the following:
(a) Search the array to find the largest element, then move it to the last position in the array.
(b) Call itself recursively to sort the first `n - 1` elements of the array.

51. What is the output of the following program? What does function `f` compute when called with a positive integer?

```
#include <stdio.h>

int f(int n);

int main(void)
{
  printf("%d\n", f(7));
}

int f(int n)
{
  int ans;

  if (n == 1)
    ans = 0;
  else
    ans = 1 + f(n / 2);

  return (ans);
}
```

52. Write and test a recursive function that returns the value of the following recursive definition:

```
f(x) = 0              if x<=0
f(x) = f(x-1) + 2          o.w.
```

53. Write a recursive function `find_sum` that calculates the sum of successive integers starting at `1` and ending at `n` (i.e., `find_sum(n) = (1 + 2 +    + (n - 1) + n).`

54. Explain the output of the following function for various values of `n`. How the function works if it is called as `f(4)`.

```
void f(int n)
{ int number;

  if (n <=1)
  { scanf("%d", number);
    printf("%d\n", number);
  }
  else
  { scanf("%d", number);
    f(n-1);
    printf("%d\n", number);
  }
}
```

55. The following program outline shows only function definitions and variable declarations.

```
int a;

void f(int b)
{ int c;
}

void g(void)
{ int d;
  {
    int e;
  }
}

int main(void)
{ int f;
}
```

For each of the following scopes, list all variable and parameter names visible in that scope:

(a) The `f` function

(b) The `g` function

(c) The block in which `e` is declared

(d) The `main` function

56. The following program outline shows only function definitions and variable declarations.

```
int b, c;

void f(void)
{ int b, d;
}

void g(int a)
{ int c;
  {
    int a, d;
  }
}

int main(void)
{ int c, d;
}
```

For each of the following scopes, list all variable and parameter names visible in that scope.
If there's more than one variable or parameter with the same name, indicate which one is visible.

(a) The `f` function

(b) The `g` function

(c) The block in which `a` and `d` are declared

(d) The `main` function