Study Questions: Set No. 7
Introduction to C
Thursday November 7, 2013

Covering:

Chapters 11 and 12

1. If `i` is a variable and `p` points to `i`, which of the following expressions are aliases for `i`?
```
(a) *p  (c) *&p    (e) *i     (g) *&i
(b) &p  (d) &p     (f) &i     (h) &*i
```

2. If `i` is an `int` variable and `p` and `q` are pointers to an `int,` which of the following assignments are legal?
```
(a) p=i;      (d) p=&q;  (g) p=*q;
(b) *p=&i;    (e) p=*&q; (h) *p=q;
(c) &p=q;     (f) p=q;        (i) *p=*q;
```

3. Show the values for `x`, `y`, and `z` that are displayed by the output of the following program.
```
#include <stdio.h>

void sum(int a, int b, int *cp);

int main(void)
{
  int x, y, z;

  x = 7; y = 2;
  printf("   x   y   z\n\n");

  sum(x, y, &z); printf("%4d%4d%4d\n", x, y, z);
  sum(y, x, &z); printf("%4d%4d%4d\n", x, y, z);
  sum(z, y, &x); printf("%4d%4d%4d\n", x, y, z);
  sum(z, z, &x); printf("%4d%4d%4d\n", x, y, z);
  sum(y, y, &y); printf("%4d%4d%4d\n", x, y, z);

  return 0;
}

void sum(int a, int b, int *cp)
{
  *cp = a + b;
}
```

4. What values of `x` and `y` are displayed by this program?
```
void f1(int *p, int y);
void f2(int *x, int *y);

int main(void)
{
  int x, y;

  f2(&x, &y);
  printf("x = %d, y = %d\n", x, y);
  return 0;
}
```

```
void f1(int *p, int y)
{ int x;

  x = 10;
  *p = 2 * x - y;
}

void f2(int *x, int *y)
{ f1(x, 7);
  f1(y, *x);
}
```

5. What values of `x` and `y` are displayed by this program?

```
void f(int x);

int main(void)
{ int x, y;

  x = 10;   y = 11;

  f(x) ;
  f(y);

  printf("x = %d, y = %d\n", x, y);
}

void f(int x)
{
  int y;

  y = x + 2;
  x *= 2;
}
```

6. What values of `x` and `y` are displayed by this program?

```
void f(int *x);

int main(void)
{ int x, y;

  x = 10;  y = 11;
  f(&x) ;
  f(&y);
  printf("x = %d, y = %d\n", x, y);
}

void f(int *x)
{
  int y;

  y = *x + 2;
  *x = 2 * *x;
}
```

7. The following function supposedly computes the sum and average of the numbers in the array `a,` which has length `n.` `avg` and `sum` point to variables that the function should modify. Unfortunately, the function contains several errors. Find and correct them.

```
void avg_sum(double a[], int n, double *avg, double *sum)
{ int i;
  sum = 0.0;
  for (i = 0 ; i < n; i++)
    sum += a[i];
  avg = sum / n;
}
```

8. Write the following function:

```
void pay_amount(int dollars, int *twenties, int *tens, int *fives, int
*toonies, int *lonnie);
```

The function determines the smallest number of $20, $10, $5, $2, and $1 bills/coins necessary to pay the amount represented by the `dollars` parameter. The `twenties` parameter points to a variable in which the function will store the number of $20 bills required. The `tens, fives, toonies` and `lonnie` parameters are similar.
To test your function, write a program that asks the user to enter an integer value for the payment amount, calls `pay_amount` to determine the smallest number of bills/coins necessary to pay the amount represented by the `dollars` parameter, and then display the values returned by the function.

9. Write a program to dispense change. The user enters the amount paid and the amount due. The program determines how many dollars, quarters, dimes, nickels, and pennies should be given as change. Write a ***function*** with four output parameters that determines the quantity of each kind of coin.

10. Write the following function:

```
void swap(int *p, int *q);
```

When passed the addresses of two variables, swap should exchange the values of the variables:

```
swap(&i, &j);/* exchanges values of i and j */
```

11. Write the following function:

```
void split_time(long total_sec, int *hr, int *min, int *sec);
```

`total_sec` is a time represented as the number of seconds since midnight range from 0 to 86399). `hr, min,` and `sec` are pointers to variables in which the function will store the equivalent time in hours (0-23), minutes (0-59), and seconds (0-59), respectively.

12. Write the following function:

```
find_two_largest(int a[], int n, int *largest, int *second_largest);
```

When passing an array `a` of length `n` to the function, it will search for the largest and second-largest elements in the array, storing them in the variables pointed to by `largest` and `second_largest,` respectively.

13. Suppose that the following declarations are in effect:

```
int a[8] = {5, 15, 34, 54, 14, 2, 52, 72};
int *p = &a[0], *q = &a[5]
```

(a) What is the value of `*(p+3)`?
(b) What is the value of `*(q-3)`?
(c) What is the value of `q - p`?
(d) What is the value of `p - q`?
(e) Is the condition `p<q` true or false?
(f) Is the condition `*p<*q` true or false?

14. Write the following function:

```
int *find_largest (int a[], int n);
```

When passed an array `a` of length `n,` the function will return a pointer to the array's largest element.

15. What will be the contents of the array `a` after the following statements are executed?

```
#define N 10

int a[N] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
int *p = &a[0], *q = &a[N-1], temp;

while (p < q)
{ temp = *p;
  *p++ _ *q;
  *q-- = temp;
}
```

16. Suppose that `high`, `low`, and `middle` are all pointer variables of the same type, and that `low` and `high` point to elements of an array. Why is the following statement illegal, and how could it be fixed?

```
middle = (low + high) / 2;
```

17. Suppose that `a` is a one-dimensional array and `p` is a pointer variable. Assuming that the assignment `p = a` has just been performed. Which of the following expressions are illegal because of mismatched types? Of the remaining expressions, which of them are true (i.e., have a non-zero value)?

```
(a) p == a[0]
(b) p == &a[0]
(c) *p == a[0]
(d) p[0] == a[0]
```

18. Write a program *using an array and an integer index only* (i.e., *without pointers*) that reads a message, then checks if it is a palindrome (the letters in the message are the same from left to right as from right to left):

```
Enter a message: He lived as a devil, eh?
Palindrome

Enter a message: Madam, I am Adam.
Not a palindrome
```

Ignore all characters that are not letters. Use integer variables to keep track of positions in the array.

19. Revise the previous program to use pointers instead of integers to keep track of positions in the array.

20. Write the following function:

```
bool search(const int a[], int n, int key);
```

where `a` is an array to be searched, `n` is the number of elements in the array, and `key` is the search key. The variable `search` should return `true` if `key` matches some element of `a`, and `false` if it doesn't. Use pointer arithmetic (not subscripting) to visit array elements.

21. Write the following function:

```
double inner_product(const double *a, const double *b, int n);
```

where `a` and `b` both point to arrays of length `n`. The function should return `a[0]×b[0] + a[1]×b[1] + ... + a[n-1]×b[n-1]`. Use pointer arithmetic (not subscripting) to visit array elements.

22. Rewrite the following function to use pointer arithmetic instead of array subscripting. (In other words, eliminate the variable `i` and all uses of the `[]` operator.) Make as few changes as possible.

```
int sum_array(const int a[] , int n)
{ int i, sum;
  sum = 0;
  for (i = 0; i < n; i++)
    sum += a[i] ;
  return sum;
}
```

23. Rewrite the following function to use pointer arithmetic instead of array subscripting. (In other words, eliminate the variable `i` and all uses of the `[]` operator.) Make as few changes as possible.

```
void store_zeros(int a[], int n)
{
  int i;

  for (i = 0; i < n; i++)
    a[i] = 0;
}
```

24. Rewrite the following function to use pointer arithmetic instead of array subscripting. (In other words, eliminate the variables `i` and `j` and all uses of the `[]` operator.) Use a single loop instead of nested loops.

```
int sum_two_dimensional_array(const int a[][LEN], int n)
{
  int i, j, sum = 0;

  for (i = 0; i < n; i++)
    for (j = 0; j < LEN; j++)
      sum += a[i][j];
  return sum;
}
```

25. Write a declaration for an 8×8 `char` array named `chess_board`. This array should be local inside the main function.

   a. Write a function that accepts as a parameter the address of an 8×8 `char` two-dimensional array and initialize it by putting 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
```

   b. Write a function that accepts as a parameter the address of an 8×8 `char` two-dimensional array and 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.

26. Write a declaration for a two-dimensional `double` array named `temperature_readings` that stores one week of temperature readings (one reading each 8 hours per day). This array should be local inside the main function. The rows of the array should represent days of the week; the columns should represent morning, afternoon, and night readings of the day.

    a. Write a function that accepts as a parameter the address of a `double` two-dimensional array, reads from the user the 21 temperature values for the week, and store them in this array.

    b. Write a function that accepts as a parameter the address of a `double` one-dimensional array and computes the average values of that array.

    c. Write a function that accepts as a parameter the address of a `double` two-dimensional array and computes the average temperature for each day of the week (averaged over the three readings of the day). Your function should call the function that you wrote in (b) to calculate this average.

    d. Write a function that accepts as a parameter the address of `double` two-dimensional array and computes the average temperature for all morning, afternoon, and night readings of the week. (averaged over all days of the week). Your function should call the function that you wrote in (b) to calculate this average.