1. break causes control to end the current loop iteration or case block and continue with the next statement. continue causes control end the current loop iteration and start the next one. break is useful when you want to stop looping, continue is useful when you want to stop the current iteration.

2. A adds the value of x to itself, effectively doubling x. B increments the value of x by one unit. A is useful only for ints and floats, B is useful also for pointers and chars.

3. A declares a function that takes a char arg and returns a pointer to an int. B declares a function that takes a pointer to a char as an argument and returns an integer.

4. A implies that x is the name of a struct and next is a name of one of its members. B implies that x is a pointer to a struct and next is a member of the struct to which x points. Use "." for structs and use "->" for pointers to structs.

5. A creates storage for 100 ints, B declares to the compiler that there is storage for 100 ints but does not create the storage.

6. A allocates 6 bytes, stores the string "betsy" in that memory and sets the symbol dog to the address of that string. B creates a pointer variable called dog and stores in that variable the address of a string the compiler stores in memory somewhere else.

7. The index variable, i, is never changed even though the pointer p is. One correction is to replace str[i] with str[i++]

8. malloc allocates memory and returns a pointer to it. The next line over-writes the pointer returned from malloc() with the address of the original string. Correction is strcpy(p, str);

9. getchar() is called twice for each iteration, checking every other character. Need to store result from getchar() in a variable.

```
10. a) month[3] = 'e'
    b)  month+3 = 482711  c) *(month+3) = 'e'

  c    t        s        month
 +---+--------+--------+----------------
 |'p'|  482709|  482709| N o v e m b e r
 +---+--------+--------+----------------
      48700    48704    48708
/* #11A
 * count diffs in strings
 */
int strdiff(char s[], char t[])
{
    int ndiffs=0, i=0, j=0;
    while( s[i] || t[j] ){
        if ( s[i] != t[j] ) ndiffs++;
        if ( s[i] ) i++;
        if ( t[j] ) j++;
    }
    return ndiffs;
}

/* #11B
 * count diffs in strings
 */
int strdiff(char *s, char *t)
{
    int ndiffs=0;
    while( *s || *t ){
        if ( *s != *t ) ndiffs++;
        if ( *s++ ) i++;
        if ( *t++ ) j++;
    }
    return ndiffs;
}

/* #11C
 * count sames in strings
 */
int strsame(char s[], char t[])
{
    return ( strlen(s) - strdiff(s,t) );
}
```

```
/* #12A  constructor*/
struct pair *newpair(char *s, char *t)
{
   struct pair *rv = malloc(sizeof(struct pair));
   if ( rv == NULL ) return NULL;
   rv->s1 = malloc(1+strlen(s));
   rv->s2 = malloc(1+strlen(t));
   if ( rv->s1 == NULL || rv->s2 == NULL )
       return NULL;
   strcpy(rv->s1,s);
   strcpy(rv->s2,t);
   rv->diffs = strdiff(s,t);
   return rv;
}

/* #12B */
/* traverse list print close matches
 */
void
print_similar(struct pair *first, int max)
{
   struct pair *ptr ;
   for( ptr = first; ptr; ptr = ptr->next )
        if ( ptr->diffs <= max )
           printf(" %d %s %s\n", ptr->diffs,
                       ptr->s1, ptr->s2);
}

/*
 * un-run-length-encode output of uniq -c
 * input is num space string newline
 */
#include <stdio.h>
#define STRLEN 1000

main()
{
   int  num, i;
   char buf[STRLEN];

   while( scanf("%d",&num) == 1 ){
      fgets(buf, STRLEN, stdin);
      while( num-- > 0 )
         printf("%s", buf);
   }
}
```