**CSE142** **26. Oct. 2000**

**Name:**

**Section:**

# Solutions to Quiz Problems

1. Complete the truth table for the expression (P && (!Q)):

| P | Q | !Q | P && (!Q) |
|---|---|---|---|
| TRUE | TRUE | FALSE | FALSE |
| TRUE | FALSE | TRUE | TRUE |
| FALSE | TRUE | FALSE | FALSE |
| FALSE | FALSE | TRUE | FALSE |

2. Fill in the conditional in the while loop so that the while loop exits only when inputChar is either 'N' or 'n':

```
char inputChar;
inputChar = 'Y';

while (inputChar != 'N' && inputChar != 'n') {
     /* Process */
     printf("Would you like to play again? ");
     scanf(" %c", &inputChar);
}
/* Final */
```

Note that this is another variety of program schema. This schema can be used in any program that needs to run at least once. This differs from the Read Until Sentinel schema that might never execute the loop body.

3. Complete the body of this function. The function should take two integers (b and k) and return the value $b^k$. Your solution must include a for-loop. (You may not use any library functions.) You can assume that b and k will always be non-negative.

Examples: `pow(2, 3)` returns `8`, and `pow(3, 2)` returns 9

```
int pow(int b, int k) {
  int result;
  for (result = 1; k > 0; k--) {
    result = result * b;
  }
  return result;
}
```

# Solutions to Quiz Section Problems

4. Using the function written in Problem 3, write a new function that takes two integers (n and b) and returns the largest integer (k) such that $b^k <= n$.

```
int gplte(int n, int b) {
  int k = 0;
  while (pow(b, k) < n) {
    k++;
  }
  return (k - 1);
}
```

5. Now for the clincher, what is the output of the following code?

```
#include <stdio.h>
void MysteryFunction(int n, int b) {
  int power, temp;
  for (power = gplte(n, b); power >= 0; power--) {
    temp = pow(b, power);
    printf("%d", n / temp);
    n = n % temp;
  }
  printf("\n");
}
int main(void) {
  MysteryFunction(13, 2);
  MysteryFunction(9 * 6, 13);
  return 0;
}
1101
42
```

It is interesting to note that MysteryFunction converts a number (n) from base 10 to base b. Thus, MysteryFunction(n, 2) converts from the format to which we are accustomed, to binary.

6. For additional practice, convert the loops you wrote in problems 3 and 4 from while-loops to for-loops, and vice versa.

7. Convert the following conditional into an equivalent conditional that does not use the negation (!) operator (consider a truth table as a sanity check):

```
!((!A) && (!(B || C)))

1) (!(!A)) || (!(!(B || C)))
2) (A) || (B || C)

or

1) !((!A) && ((!B) && (!C)))
2) !((!A) && (!B) && (!C))
3) ((!(!A)) || (!(!B)) || (!(!C)))
4) (A || (B || C))
```

8. What are the values of a, b and c after executing this code?

```
#define TRUE 1
#define FALSE 0
int a, b, c;
a = TRUE;
b = 6;
c = 8;
while ((!a) || ((b > 2) && (c <= 12))) {
  if (a)
    c = b * 2;
  else
    b = c - 7;
  a = (!a);
}

a = 1
b = -1
c = 6
```