Indian Institute of Technology Mandi

# IC150: Computation for Engineering

## Tutorial 2 Model Solutions

1) Fill in the blanks

   (a) A pointer is a variable that contains as its value the __address__of another variable.

   (b) The elements of an array are related by the fact that they _ have the same datatype.

   (c) An m-by-n array contains _m__rows, _n_ columns, and ___m×n__ elements.

   (d) Assume a is an array and p is a pointer. The assignment involving a and p that is not valid in C is ______a = p_________

2) Use a single-subscripted array to solve the following problem. Read in 100 numbers, each of which is between 0 and 20, inclusive. As each number is read, print it only if it is not a duplicate of a number already read. Provide for the "worst case" in which all 100 numbers are different. Use the smallest possible array to solve this problem. Write pseudo-code (not C code).

   Ans 1. Strategy: Store the unique numbers in an array A. At any time, c is the cardinality of A. For each number read, if it is not already in A, add it to the end of A and print it.

```
1.  let c ← 0
2.  for i from 1 to 100 do
2.1.    read next input into n
2.2     for j from 1 to c do
2.2.1           if $A_j$ = n skip to next iteration of loop 2
2.3     increment c
2.4     $A_c$ ← n
2.5     print n
```

   Ans 2. Strategy: Use a 21-element array Seen[] to keep track of the numbers that have been seen. For each number read, if it is not already in Seen[], set the corresponding element to True.

```
1.  for i from 0 to 20 do
1.1     Seen[i] = 0 false
2.  for i from 1 to 100 do
2.1.    read next input into n
2.2     if Seen[i] is false
2.2.1           print n
2.2.2           Seen[i] = true
```

   Note: Ans 2 is better if the range of the input numbers is smaller than the number of inputs. Ans 1 is good if the range of input numbers is very large, eg. for input numbers from 0-100,000.

3) Label the elements of 3-by-5 double-subscripted array sales to indicate the order in which they are set to zero by the following program segment:

```
for(row=0; row<3; row++)
   for(column=0; column<5; column++)
```

```
        sales[row][column]=0;
Ans. sales[0][0], sales[1][0], sales[2][0],
     sales[0][1], sales[1][1], sales[2][1],
     sales[0][2], sales[1][2], sales[2][2],
     sales[0][3], sales[1][3], sales[2][3],
     sales[0][4], sales[1][4], sales[2][4]
```

4) (a) Write a *recursive* function `void PrintVector(int v[], int size)` to print the `size` elements of `v` in order on one line, separated by space. Each call to `PrintVector` should print only one element and call itself recursively.

```
Ans. PrintVector(int v[], int size)
      {
        if (size > 0)
          {
            printf("%d ", v[0]);
            PrintVector(v+1, size-1);
          }
       }
```

(b) Write an *iterative* function `void PrintMatrix(int m[][MAX], int rows, int cols)` to print the elements of `m` on `rows` lines. `PrintMatrix()` should use `PrintVector()` to print each row.

```
Ans. void PrintMatrix(int m[][MAX], int rows, int cols)
      {
        int r;
          for (r=0; r<rows; r++)
            {
               PrintVector(a[r], cols);
               printf("\n");
            }
       }
```

5) Answer each of the following. Assume that unsigned integers are stored in 2 bytes, and that the starting address of the array is at location 1002500 in memory.

(a) Declare an array of type `unsigned int` called `values` with 5 elements, and initialise the elements to the even integers from 2 to 10. Assume the symbolic constant `SIZE` has been defined as `5`.

```
Ans.   #define SIZE 5
       unsigned int values[SIZE] = {2,4,6,8,10};
```

(b) Declare the pointer `vPtr` that points to an object of type `unsigned int`.

```
Ans. unsigned int *vPtr;
```

(c) Print the elements of array `values` using array subscript notation. Use a `for` structure and assume integer control variable `i` has been declared.

```
Ans. for(i=0; i<SIZE; i++)
       printf(“%d ”, values[i]);
```

(d) Give two separate statements that assign the starting address of array `values` to pointer variable `vPtr`.

Ans.
```
 vPtr = values;
vPtr = &values[0];
```

(e) Print the elements of array `values` using pointer/offset notation.

Ans.
```
for(i=0; i<SIZE; i++)
   printf(“%d”, *(vPtr + i));
```

(f) What address is referenced by `vPtr + 3`? What value is stored at that location.

Ans. 1002506, 8

6) Hand simulate the function below for the calls `n = Mystery(“”)` and `n = Mystery(“Quiz 2”)`. What does this function do?

```
int Mystery(char *s)
  {
    for(int x=0; *s != '\0'; s++ )
      ++x;
    return x;
  }
```

**Ans.** Hand simulation of `Mystery(“”)`

| **Line** | **s** | **x** |
|---|---|---|
| On entry to for loop | \0 | 0 |
| At end of Mystery() | \0 | 0 |

Hand simulation of `Mystery(“Quiz 2”)`

| **Line** | **s** | **x** |
|---|---|---|
| On entry to for loop | Quiz 2\0 | 0 |
| End of iteration 1 | uiz 2\0 | 1 |
| End of iteration 2 | iz 2\0 | 2 |
| End of iteration 3 | z 2\0 | 3 |
| End of iteration 4 | 2\0 | 4 |
| End of iteration 5 | 2\0 | 5 |
| End of iteration 6 | \0 | 6 |
| At end of Mystery() | \0 | 6 |

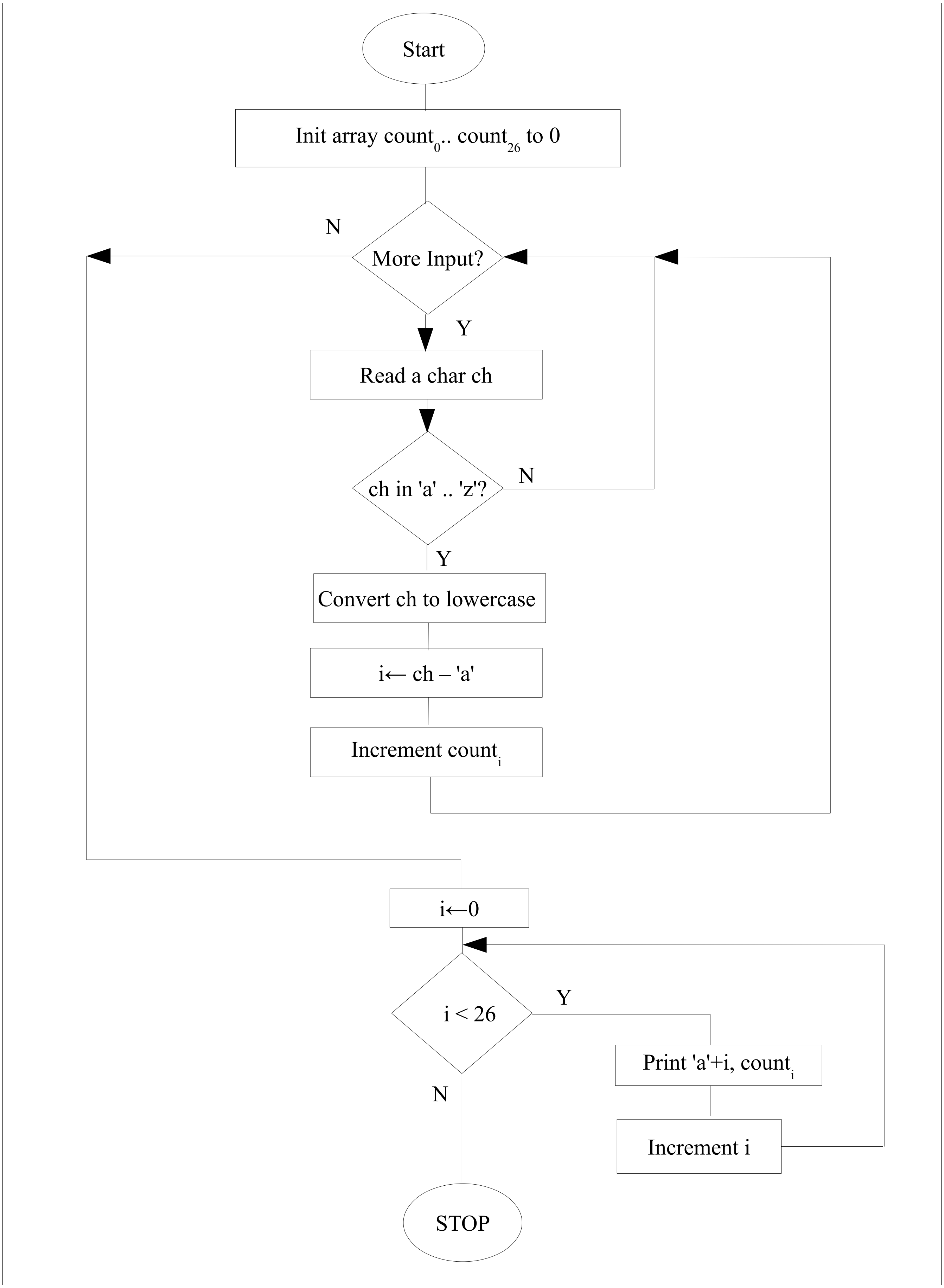
This function returns the length of string s passed as its argument, not including the terminating '\0'.

7) Design a program that reads several lines of text from the keyboard and prints a table indicating the number of occurrences of each letter of the alphabet in the text. For example, the phrase

```
To be, or not to be: that is the question
```

contains one “a”, two “b”, no “c”, etc. Draw a neat flowchart for the program.

Ans: We assume that characters are represented in ASCII or equivalent such that C-style arithmetic on characters is possible.


Start
Init array count$_0$.. count$_{26}$ to 0
N
More Input?
Y
Read a char ch
ch in 'a' .. 'z'?
N
Y
Convert ch to lowercase
i← ch – 'a'
Increment count$_i$
i←0
i < 26
Y
Print 'a'+i, count$_i$
N
Increment i
STOP

8) Assume that the first two digits of a mobile number identify the operator, eg. 94 is BSNL, 98 is Airtel, 93 is Reliance, etc. Given a mobile number in the format xxyyy-yyyyy, it is desired to print it in the format <operator name>-yyy-yyyyy. Eg. 94180-43219 should be printed as BSNL-180-43219. Devise an algorithm to read a phone number and print it out in the desired format. Decide on the arrays and other variables necessary and write pseudo-code.

```
Assume that operator ids range from 80-99.
Let opName be an array of 20 strings
Init $opName_i$ to the operator name for prefix i+80.  Eg, $opName_{14}$ ← “BSNL”
Read input as 2 5-digit integers into p1 and p2
opIndex ← (p1 / 1000) – 80
p1 ← p1 mod 1000
print concatenation of $opName_{opIndex}$, “-”, p2
```