Name: Section: Rollno:

ESC101A: Fundamentals of Computing(Major Quiz 2-A)

28th October, 2014

Total Number of Pages: 5 Total Points 47

Question	Points	Score
1	15	
2	20	
3	12	
Total:	47	

Instructions

1. Read these instructions carefully.
2. Write you name, section and roll number on all the pages of the answer book.
3. Write the answers cleanly in the space provided. There is space left on the back of the answer book for rough work.
4. Do not exchange question books or change the seat after obtaining question paper.
5. Using pens (blue/black ink) and not pencils. Do not use red pens for answering.
6. Even if no answers are written, the answer book has to be returned back with name and roll number written.

Helpful hints

1. The questions are *not* arranged according to the increasing order of difficulty. Do a quick first round where you answer the easy ones and leave the difficult ones of the subsequent rounds.
2. For fill in the blanks type of questions, read the comments in the code. They usually have helpful remarks.

Name: Section: Rollno:

Question 1. (15 points) What is the output of the following program?

```
#include<stdio.h>
#include<stdlib.h>
/* some useful characters */
#define NEWLINE '\n'    /* enter OR newline   */
#define SPACE   ' '     /* space              */
#define FS      '/'     /* forward slash      */
#define BS      '\\'    /* single backslash \ */
#define VB      '|'     /* vertical bar       */
#define HB      '-'     /* horizontal bar     */
#define CB      '+'     /* cross bar          */

void genD(char** D, int n)
{
    int i, j;

    for (i=0; i < n/2; i++) {
        for (j=0; j < n/2; j++)   D[i][j] = BS;
        for (j=n/2+1; j < n; j++) D[i][j] = FS;
    }

    for (i=n/2+1; i <n; i++) {
        for (j=0; j < n/2; j++)   D[i][j] = FS;
        for (j=n/2+1; j < n; j++) D[i][j] = BS;
    }

    for (i = 0; i < n; i++) {
        D[i][n/2] = VB;
        D[n/2][i] = HB;
    }
    D[n/2][n/2] = CB;
}

int main()
{
    const int n = 5;      int i, j;
    char** D = (char**) malloc(n*sizeof(char*));
    for (i=0; i < n; i++)
        D[i] = (char*) malloc(n*sizeof(char));

    genD(D, n); /* matrix populated here */

    for (i=0; i<n; i++)
        for (j=0; j<n; j++) {
            /* PRINT the matrix */
            printf("%c%c", D[i][j], j==n-1 ? NEWLINE : SPACE);
        }

    return 0;
}
```

Output (Grid provided to help in aligning the output):

Name: Section: Rollno:

Question 2. (20 points) What is the output of the following program for the given input? One example is provided for you.

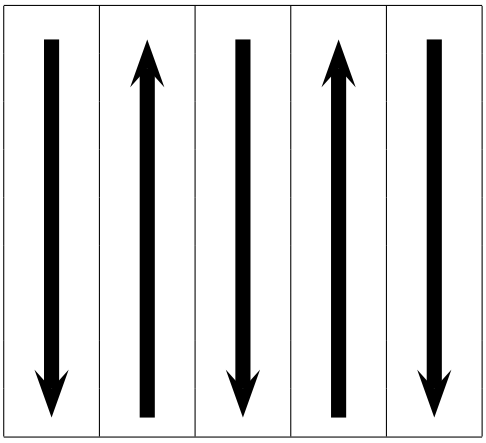
```
1  #include<stdio.h>
2
3  int mcNO(int n) {
4    static int callNum = 1;
5    printf("%3d: mcNO(%3d)\n", callNum, n); /* PRINTF HERE */
6    callNum++;
7
8    if (n > 100)
9      return n - 10;
10   else
11     return mcNO( mcNO (n + 11) );
12
13 }
14
15 int main() {
16   int N;
17   scanf("%d", &N);
18
19   printf("Result is %d\n", mcNO(N));      /* PRINTF HERE */
20
21   return 0;
22 }
```

Input: **100**	Input: **97**
Output: 1: mcNO(100) 2: mcNO(111) 3: mcNO(101) Result is 91	**Output:**

Name: Section: Rollno:

Question 3. (12 points)

The following partially filled function takes as input a one-dimensional array A of length len^2. It creates and populates a 2-dimensional array B of dimension $len \times len$ using elements of A. The array B is filled column-wise in a snake-like fashion, starting from first row of first column, i.e.



For example, if array A is: | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | then matrix B will be:

1	6	7
2	5	8
3	4	9

Fill in the missing blanks and complete the program. There are **7** blanks that you need to fill.

```
1  #include<stdlib.h>
2
3  int** snake(int* A, int len)
4  {
5      int **B;
6      int i, j;
7      B = _____ malloc (_____);
8      for (i = 0; i < _____; i++)
9          B[i] = _____ malloc (_____);
10
11     /* fill the matrix in column-wise fashion */
12     for (j = 0; j < len; j++) {
13         for (i = 0; i < len; i++)
14             if (j%2 == 0) { /* 0, 2, 4, .. one direction */
15                 B[i][j] = _____;
16             }
17             else { /* 1, 3, ... other direction */
18                 B[i][j] = _____;
19             }
20     }
21
22     return B;
23 }
```