ESC101N: Fundamentals of Computing
Mid-sem 2
2010-11 1st semester

Instructor: Arnab Bhattacharya

2:00-3:00pm, 8th October, 2010

Instructions

1. Please write your name, roll number and section below.
2. Please write your roll number and section in each page of the question paper.
3. Please write your answers in the boxes provided for them.
4. Please hand over the paper after the exam.
5. This question paper contains 4 questions in 8 pages.

Name:

Roll Number:

Section:

Question	1	2	3	4	Total
Marks	15	11	10	14	50
Score					

[15] 1. In the following program, `s` is a 4x4 matrix of integers. Assume that the user will input integers only between 0 and 3. The program checks whether each row of the matrix contains each of the four integers 0, 1, 2 and 3 *exactly once*. If so, it prints the message "Matrix is perfect". Otherwise, it shows which row has a number occurring more than once or which row has a number not occurring at all.

For this, it uses an auxiliary array `a` that maintains the count of each of the numbers in a row. Later, it checks whether the counts of each element of `a` is exactly 1. Otherwise, there is an error.

Complete the code segment below by appropriately filling in the blanks ??1?? to ??10??. (Note that ??6??, ??7?? and ??8?? appear twice in lines 29, 35 and 30, 36.)

```
 1 #include <stdio.h>
 2
 3 int main()
 4 {
 5     int i, j;
 6     int m;
 7     int a[4];
 8     int good = 1;
 9     int s[4][4];
10
11     for (i = 0; i < 4; i++)
12         for (j = 0; j < 4; j++)
13             scanf("%d", &s[i][j]);
14
15     for (i = 0; ??1??; i++)
16     {
17         for (m = 0; m < 4; m++)
18             a[m] = ??2??;
19
20         for (j = 0; ??3??; j++)
21         {
22             ??4??
23         }
24
25         for (m = 0; m < 4; m++)
26         {
27             if (??5??)
28             {
29                 printf("%d found more than once in row %d\n", ??6??,
                       ??7??);
30                 good = ??8??;
31             }
32
33             if (??9??)
34             {
35                 printf("%d not found in row %d\n", ??6??, ??7??);
36                 good = ??8??;
37             }
38         }
39     }
```

```

    if (??10??)
        printf("Matrix is perfect\n");
}
```

Answer:
??1?? : `i < 4` (1 mark)
??2?? : `0` (1 mark)
??3?? : `j < 4` (1 mark)
??4?? : `a[s[i][j]]++;` or any equivalent code (4 marks, 1 mark each for i, j, s[i][j] and ++)
??5?? : `a[m] > 1` (2 marks)
??6?? : `m` (1 mark)
??7?? : `i` (1 mark)
??8?? : `0` (1 mark)
??9?? : `a[m] == 0` (2 marks)
??10??: `good == 1` or `good` (1 mark)

[11] 2. The function `substring` in the following program tries to find the *last* occurrence of the sequence of numbers represented by `q` in another sequence of numbers represented by `s`. Assume that both the sequences are of positive integers. The sequence ends with the number 0. Note that this is the *only way* to signal the end of a sequence. The array sizes declared in the `main` function specify the largest sizes.

The function returns -1 if the sequence `q` is not found in `s`. Otherwise, it returns the *last* index of `s` that matches with `q`. The variable `t` stores the running index of `q` matching with `s`. Note that the conditions in the `for` loops at lines 8 and 13 are blank.

Fill in the blanks ??1?? to ??5?? appropriately to complete the program.

```
 1 #include <stdio.h>
 2
 3 int substring(int s[], int q[])
 4 {
 5     int i, j, t;
 6     int p = -1;
 7
 8     for (i = 0; ; i++)
 9     {
10         if (s[i] ??1??)
11             break;
12         t = -1;
13         for (j = 0; ; j++)
14         {
15             if (q[j] ??2??)
16             {
17                 t = i;
18                 break;
19             }
20             if (??3??)
21             {
22                 t = -1;
23                 break;
24             }
25         }
26         if (t != -1)
27             p = ??4??;
28     }
29
30     return p;
31 }
32
33 int main()
34 {
35     int string[100], query[50];
36     int i, j, index;
37
38     for (i = 0; i < 100; i++)
39     {
40         scanf("%d", &string[i]);
41         if (string[i] == 0)
```

```
                break;
    }
    for (j = 0; j < 50; j++)
    {
        scanf("%d", &query[j]);
        if (query[j] == ??5??)
            break;
    }

    index = substring(string, query);
}
```

Answer:

??1??: `== 0` (1 mark)
??2??: `== 0` (2 marks)
??3??: `(s[i + j] == 0) || (s[i + j] != q[j])` (6 marks, 3 marks for == 0 condition, 2 marks for != condition and 1 mark for ——)
??4??: `t` or `i` (1 mark)
??5??: `0` (1 mark)

[10] 3. The function `search` in the following program finds the occurrence of first 1 in an array of 0s and 1s. If the array contains only 0, it should return -1. If the function encounters any integer other than 0 or 1 before it finds the first 1, it should return -2. Otherwise, it should return the index of the first 1. Note that the function is *recursive*.

Fill in the blanks ??1?? to ??4?? appropriately to complete the program.

```
#include <stdio.h>
int search(int a[], int i, int n)
{
    if (??1??)
        return -1;
    else if (??2??)
        return i;
    else if (a[i] == 0)
        return search(??3??);
    else
        return ??4??;
}

int main()
{
    int i, s;
    int a[4];

    for (i = 0; i < 4; i++)
        scanf("%d", &a[i]);

    s = search(a, 0, 4);
    printf("search returns %d\n", s);
}
```

Answer:

??1??: `i == n` (2 marks)
??2??: `a[i] == 1` (2 marks)
??3??: `a, i + 1, n` (1 + 2 + 2 marks)
??4??: `-2` (1 mark)

4. Study the following program. *Replace* the number within <> in line 31 by the *last digit of your roll number*. For example, if your roll number is 10678, assume that the statement in line 31 is
   `*a = 8 % 3 + 1;`

```
 1 #include <stdio.h>
 2
 3 void f(double *d, int s)
 4 {
 5     double *e;
 6     int i = 0;
 7     double n = 0;
 8     int j;
 9
10     e = d - 1;
11
12     while (i < s)
13     {
14         n = n + (*d);
15         printf("n = %lf\n", n);
16         d++;
17         i++;
18     }
19
20     j = d - 1 - e;
21     printf("j = %d\n", j);
22
23     *e = n / j;
24 }
25
26 int main()
27 {
28     double a[5];
29     int i;
30
31     *a = <last digit of your roll number> % 3 + 1;
32     for (i = 1; i < 5; i++)
33         *(a + i) = a[0] + 4.0 * i;
34
35     printf("a[0] = %lf\n", a[0]);
36
37     f(a + 1, 4);
38
39     printf("*a = %lf\n", *a);
40 }
```

[9] (a) What is the output of the following program? In case any of the output is unknown or cannot be determined, please clearly say that.

[5] (b) What does the program do?

Answer:

(a) (-3 for not replacing by last digit of roll number)

a[0] =	1	2	3	(1 mark)
n =	5	6	7	(1 mark)
n =	14	16	18	(1 mark)
n =	27	30	33	(1 mark)
n =	44	48	52	(1 mark)
j =	4	4	4	(2 marks)
*a =	11	12	13	(2 marks)

(b) The program computes the average of the 4 numbers stored at `a[1]` to `a[4]` (3 marks, 1 mark for average of 5 numbers). It then stores the average in a[0] (2 marks, 1 mark for mentioning storage only or wrongly).