Programming and Data Structures
Mid-Semester - Solutions to Sample Questions
Dept. of Computer Science and Engg.
IIT Kharagpur
Spring 2015-16

February 15, 2016

1. Tick the correct options.

   (a) Consider the following program segment.

```
main(){
    int m, n, temp;
    printf("Give integers m and n: ");
    scanf(":%d, %d", &m, &n);
    do{
        if(m < n) n = n-m;
        if(n < m) { temp = m; m = n; n = temp; }
        if(m > n) printf( "!");
    } while(m < n);
}
```

   Which of the following is true for the number of '!'s printed by the program?

   i. For some values of 'm' and 'n', exactly one '!' is printed.
   ii. For all values of 'm' and 'n', no '!' is printer.
   iii. For all values of 'm' and 'n', the number of '!'s printed is either 1 or a positive even number.
   iv. For all values of 'm' and 'n', the number of '!'s printed is a positive odd number.

   Solution: [1(a)ii] The second `if` will make n > m, if it isn't already so. The third `if` will therefore always fail and no '!' will be printed.

   (b) Given the following definitions, what is the value of b[1][1]?

```
int b[3][2]={{1,2},{3,4},{5}};
```

   i. 3
   ii. 4
   iii. 5
   iv. None of the above.

Solution: [1(b)ii]

2. A number is super prime if it is prime and all the numbers obtained by slicing one or more of its right-most digits is also prime. For example: 7331 is prime, 733 is prime, 73 is prime and 7 is also a prime. Hence, the number 7331 is a super prime.

   (a) First , complete the following function that checks if a number is a prime.
   Solution

```
int check_prime(int x){
    int k, j;
    int flag = 1;

    if (x == 1) return 0;
    else if (x == 2) return 1;

    for(j=2; j<=x-1; j++){
                k = x % j;     // Is x divisible by j?
                if (k == 0){
                        flag = 0; // Assuming 1 is not prime,
                                  // 0 indicates this.
                        break;
                }
    }
    return flag;
}
```

   (b) Use the `check_prime()` function to check if a positive integer `i` is a super prime.
   Solution

```
int check_super(int i){
        int temp;
        int flag = 1;

        temp = i;
        while(temp > 0){
                flag = check_prime(temp);
                temp = temp/10;
                if(flag == 0) break;
        }
        return flage;
}
```

3. Let a_1, a_2, ..., a_n be a sequence of positive integers. An increasing subsequence of length l is a contiguous block a_i, a_{i+1},..., $a_{i+l?1}$ satisfying $a_i \leq a_{i+1} \leq \ldots \leq a_{i+l-1}$.

   Write a C program to read a sequence of positive integers and to print the length of the longest increasing subsequence in it. In order to terminate the sequence, the user should enter zero or a negative value. Your program **must contain only one loop**. Both scanning the next integer and processing the scanned integer should be done in that loop. Write no functions other than main(). **Do not use any array**.

Here is a sample run of your program.

```
Enter an integer:  9
Enter an integer:  2
Enter an integer:  6
Enter an integer:  8
Enter an integer:  5
Enter an integer:  7
Enter an integer:  -1
Length of the longest increasing subsequence = 3
```

Solution

```
#include <stdio.h>
int main ()
{
        int a, prev = -1, runningmaxlen = 0, maxlen = 0;
        while (1)
        {
                printf("Enter an integer: "); scanf("%d", &a);
                if (a <= 0)
                {
                        if (runningmaxlen > maxlen)
                                maxlen = runningmaxlen;
                        break;
                }
                if (a >= prev)
                {
                        ++runningmaxlen;
                }
                else
                {
                        if (runningmaxlen > maxlen)
                                maxlen = runningmaxlen;

                        runningmaxlen = 1;
                }
                prev = a;
        }
        printf("Length of the LIS = %d\n", maxlen);
}
```

4. Write a function `squeeze(s1,s2)` that deletes each character in the string *s1* that matches any character in the string *s2*.

```
void squeeze(char s1[], char s2[])
{
    int i, j, k;
    int instr2;

    for (i = j = 0; s1[i] != '\0'; i++) {
        instr2 = 0;
        for (k = 0; s2[k] != '\0' && !instr2; k++)
            if (s2[k] == s1[i])
                instr2 = 1;
        if (!instr2)
            s1[j++] = s1[i];
    }
    s1[j] = '\0';
}
```

5. You are given an array. You have to write a program that will print all the leaders in the array. An element is leader if it is greater than all the elements to its right side. And the rightmost element is always a leader.

   For example for array `6, 7, 4, 3, 5, 2`, leaders are 7, 5 and 2.

   Solution

```
void printLeaders(int arr[], int size)
{
    int max_from_right =  arr[size-1];

    /* Rightmost element is always leader */
    printf("%d ", max_from_right);

    for (int i = size-2; i >= 0; i--)
    {
        if (max_from_right < arr[i])
        {
            max_from_right = arr[i];
            printf("%d ", max_from_right);
        }
    }
}
```

6. What will be the output of the following C code?

```
int main()
{
    int i=0;
    while ( +(+i--) != 0)
        i-=i++;
    printf("%d",i);
    return 0;
}
```

Solution

Output: -1

Let us first take the condition of while loop. There are several operator there. Unary + operator doesn?t do anything. So the simplified condition becomes (i?) != 0. So i will be compared with 0 and then decremented no matter whether condition is true or false. Since i is initialized to 0, the condition of while will be false at the first iteration itself but i will be decremented to -1. The body of while loop will not be executed. And printf will print -1.

7. Print numbers from 1 to 100, without using loop.
Solution

```
void printNos(int n)
{
        if(n > 0)
        {
            printNos(n-1);
            printf("%d ",  n);
        }
        return;
}

void main()
{
        printNos(100);
}
```

8. Write a program to print the following :

```
A B C D E F G G F E D C B A
A B C D E F F E D C B A
A B C D E E D C B A
A B C D D C B A
A B C C B A
A B B A
A A
```

Solution

```
#include<stdio.h>
#include<conio.h>
int main( )
{
        int r,c,askey;
        clrscr( );

        for(r = 7; r >= 1; r-- )
        {
                askey=65;
                for(c = 1; c <= r; c++ )
                        printf("%c", askey++ );
                askey--;
                for(c=r; c>=1; c-- )
                        printf("%c", askey--);

                printf("\n");
        }
        getch();
}
```

9. Reverse a string in C without using reverse function.

Solution

```
#include<stdio.h>
int main()
{
        char str[50];
        char rev[50];
        int i=-1,j=0;

        printf("Enter any string : ");
        scanf("%s",str);

        while(str[++i]!='\0');

        while(i>=0)
                rev[j++] = str[--i];

        rev[j]='\0';

        printf("Reverse of string is : %s",rev);

        return 0;
}
```

10. Write a program to check whether two strings are anagrams or not (string is assumed to consist of alphabets only). Two words are said to be anagrams of each other if the letters from one word can be rearranged to form the other word. For example "abc" and "cab" are anagram strings, here every character 'a', 'b' and 'c' occur only one time in both strings.

Solution

```
#include <stdio.h>

int check_anagram(char [], char []);

int main()
{
   char a[100], b[100];
   int flag;

   printf("Enter first string\n");
   gets(a);

   printf("Enter second string\n");
   gets(b);

   flag = check_anagram(a, b);

   if (flag == 1)
      printf("\"%s\" and \"%s\" are anagrams.\n", a, b);
   else
      printf("\"%s\" and \"%s\" are not anagrams.\n", a, b);

   return 0;
}

int check_anagram(char a[], char b[])
{
   int first[26] = {0}, second[26] = {0}, c = 0;

   while (a[c] != '\0')
   {
      first[a[c]-'a']++;
      c++;
   }

   c = 0;

   while (b[c] != '\0')
   {
      second[b[c]-'a']++;
      c++;
   }
```

```
for (c = 0; c < 26; c++)
{
   if (first[c] != second[c])
      return 0;
}

return 1;
```