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

Instructor: Arnab Bhattacharya

8:00-11:00am, 15th November, 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. Please switch off your mobile phone and deposit it to the front desk. Your mobile will be confiscated if it is with you and marks will be deducted.
6. This question paper contains 8 questions in 13 pages.

Name:

Roll Number:

Section:

Question	1	2	3	4	5	6	7	8	Total
Marks	43	10	23	16	16	13	14	15	150
Score									

[14] 1. (a) Find the errors in the following program that would be shown during compilation time or runtime. Indicate the line number, the error and when it is shown (i.e., whether at compile time or runtime).

```
 1 #include <stdio.h>
 2 void f(int p, int q)
 3 {
 4   return p + q;
 5 }
 6
 7 int main()
 8 {
 9   int i, n, a[50][50];
10   int *p, *q;
11   char *s;
12   scanf(``%d'' &n);
13   for (i = 0; i < n; i++)
14     for (j = 0; j < n; j++)
15       a[i][j] = f(i, j);
16   p = *(a + 2);
17   q = *(*(a + 2) + 2);
18   printf(``%u\n'', p + q);
19   s[0] = '\0';
20 }
```

[5] (b) What is the output of the following program? If there is an error, mention the line where it occurs and its cause. Ignore the error(s) and write the rest of the output.

```
 1 #include <stdio.h>
 2 int main()
 3 {
 4   int x = 2, y = 3;
 5   {
 6     int x = 5, z = 4;
 7     {
 8       int w = 2;
 9       int z = 6;
10       printf("x + w = %d\n", x + w);
11       printf("y + w = %d\n", y + w);
12     }
13     printf("y + z = %d\n", y + z);
14     printf("z + w = %d\n", z + w);
15   }
16   printf("x + y = %d\n", x + y);
17 }
```

[3] (c) What are the values of `x`, `y` and `z` after executing the following code?

```
int a = 2, b = 4;
```

```
float x, y, z;
x = a/b;
y = (float)(a/b);
z = (float)a/b;
```

[14] (d) For each of the following expressions, mention whether it is valid. Evaluate it if it is valid and state the reason if it is invalid.

```
int i = 10, j = 5, k = 1, m;
float w = 7;
```

i. `10 - 20 / 8 * 5 - -5`
ii. `i / j + k * m`
iii. `j / i * i + j % i`
iv. `(w % 2 == 0) || (w % 3 == 1)`
v. `(i % j) && (j / (k - 1))`
vi. `(‘k’ - ‘i’ + j) * k - i + j`

[4] (e) Define a structure that contains three fields: an integer, an array of doubles of size 10 and a pointer to a character.

[3] (f) Find the asymptotic upper bounds of running time ($O(.)$ notation) for the following functions:
i. $f(n) = 10n^2 + (1/10)n^3$
ii. $g(n) = (1/9)n \log n + 1000n + 2$
iii. $h(n) = n^{3/2} + n^{2/3}$

Answer:
(a)
1. Line 2 or 4 or 15: `f` returns an integer when return type is `void`; compile [3]
2. Line 12: comma missing in `scanf`; compile [1]
3. Lines 14, 15: `j` not declared [2]; compile
4. Line 17: type mismatch for `q`: `int *` required instead of `int`; compile [3]
5. Line 18: `+` on pointers is invalid; compile [2]
6. Line 19: `s[0]` is assigned without allocating space for pointer `s`; runtime [3]

Mentioning any spurious error will cause marks to be deducted.
(b)
x + w = 7
y + w = 5
y + z = 7
Error in line 15 as `w` is not defined [0.5 for not mentioning the reason for error.]
x + y = 5
Identifying spurious errors causes marks to be deducted.
(c)
x = 0.00000

y = 0.00000
z = 0.50000
(d)
i. Valid; 5 [2]
ii. Valid; garbage or random value [3] (1 mark for any integer, 2 for garbage)
iii. Valid; 5 [2]
iv. Invalid; % operates only on integers [2]
v. Valid; 0 [3, 1 + 2]
vi. Valid; 2 [2]
(e)

```
struct S
{
  int r;
  double m[10];
  char *n;
};
```

1 for correct syntax for `struct` and 1 for each of the variables.
(f)
i. $f(n) = O(n^3)$
ii. $g(n) = O(n \log n)$
iii. $h(n) = O(n^{3/2})$
0.5 for any other higher order that is correct, e.g., $O(n^5)$.

[6] 2. (a) The following code segment assumes that the beginning of a linked list is maintained by `head`. The nodes in the linked list contain a single value of type integer as the variable `data` and a pointer `next` to the same type of structure. Complete the following code segment such that pointer `p` points to the node whose `data` value is 9. If no such node exists, `p` must be `NULL`.

```
p = ??1??;
while (??2??)
  p = ??3??;
```

[4] (b) What is the asymptotic upper bound of the average time complexity of the above program in O notation? Assume that the linked list contains n nodes. Explain. What is the time complexity of the best case for searching? When does it happen?

Answer:
(a)
??1??: `head` [1]
??2??: `(p != NULL) && (p->data != 9)` [3]
??3??: `p->next` [2] (1 for `p.next`)

(b)
Average: $O(n)$ since on average $n/2$ nodes will be searched.
Best: $O(1)$ when the first node is the answer.

[5] 3. (a) The following function `parity` finds the number of 1's in an array. If it is even, it returns a 1, and if it is odd, it returns a 0. Fill in the blanks to complete the code segment.

```
int parity(int a[], int n)
{
    int i, c = 0;
    for (i = 0; i < n; i++)
        if (??1??)
            ??2??;
    return ??3??;
}
```

[10] (b) What is the output of the following program?

```
#include <stdio.h>
int main()
{
  int i;
  for (i = 3; i > -4; i = i - 2)
  {
    printf("%d %d\n", 5 - i + 1, 30 - 5 * i);
    if (i < 0)
      i = i + 1;
  }
}
```

[8] (c) Convert the following code segment using `for` loops into an *equivalent* code that does not use `for` loops, but instead uses `while` loops.

```
int i, a[100][100];
for (i = 1; i < 25; i = i + 2)
{
  int j = 0;
  for (j = 1; j <= 25; j++)
    a[i - 1][j - 1] = i;
}
```

Complete the code segment given below. You can write *only one simple* statement for each blank. You cannot put in a block of code for a blank.

```
int i, a[100][100];
??1??
```

```
while (i < 25)
{
  ??2??
  while (j <= 25)
    ??3??
  ??4??
}
```

Answer:

(a)

??1??: `a[i] == 1` [2]

??2??: `c++` [1]

??3??: `1 - c % 2` [2]

(b)

3 15
5 25
7 35
8 40
9 45

(c)

??1??: `i = 1;` [1]

??2??: `int j = 1;` [2]

??3??: `a[i][j++ - 1] = i;` [4] (2 for only j, 1 for using block of code)

??4??: `i = i + 2;` [1]

[16] 4. The following program inputs a positive integer `n` as a *command line argument*. It also subsequently reads a file name as the second command line argument, reads the first `n` words from the file, and prints them. If there are less than `n` words in the first file, the program finds the end of file before reading `n` words, and prints only those words. A word is separated from another word by whitespace, and is at most 20 characters long. Fill in the blanks to complete the code.

```
#include <stdio.h>
int main(??1??)
{
    FILE *fp;
    char word[20];
    int n, i;
    fp = ??2??;
    n = ??3??;
    for (i = 0; i < n; i++)
    {
        if (??4??)
```

```
                break;
            fscanf(??5??);
            printf(??6??);
        }
        fclose(??7??);
    }
```

Answer:

??1??: `int argc, char *argv[]` [3, 1 for `argc`, 2 for `argv`, -0.5 for omitting `*` or `[]`]
??2??: `fopen(argv[2], ‘‘r’’)` [3]
??3??: `atoi(argv[1])` [2]
??4??: `feof(fp)` [2]
??5??: `fp, ‘‘%s’’, word` [3]
??6??: `‘‘%s ’’, word` [2]
??7??: `fp` [1]

[8] 5. (a) Define a *recursive* function that takes a string and prints it in reverse. The function should take only one parameter: the pointer to the start of the string. For example, if the string "abcd ef" is passed to the function, it should print "fe dcba".

[8] (b) What is the output of the following program? What does the program store in `w` with respect to `s` and `t`?

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int main()
{
    char *s = "abc";
    char *t = "de";
    int n = strlen(s), m = 1 + strlen(t);
    int l = n * m + 1;
    char *r = (char *)malloc(l * sizeof(char));
    char *p, *q, *w;
    q = s;
    w = r;
    while (*q != '\0')
    {
        *r = *q;
        r++;
        p = t;
        while (*p != '\0')
        {
            *r = *p;
            r++;
            p++;
        }
        q++;
    }
    *r = '\0';
    printf("%s %s %s", s, t, w);
}
```

Answer:

(a)

```
void reverse(char *c) [1]
{
  if (*c == '\0') [1]
    return; [1]
  else
  {
    reverse(c + 1); [3]
    printf("%c", *c); [2]
  }
}
```

If the program does not use recursion, one mark if the prototype is correct (`void f(char *s)`). If the return type is anything other than `void`, no marks are given.

If the program uses recursion but modifies the original string, and is otherwise all correct, four marks are awarded. One mark is deducted for each error such as function prototype not being correct.

If the program uses recursion and is all correct including no modification in the original string, 8 marks are given. One mark is deducted for each error condition such as function prototype not being correct.

If the program does not check for the right base case (i.e., string being all empty with string length being zero), one mark is deducted.

(b) “abc” “de” “adebdecde” [1 + 1 + 3,]

The program inserts the string `t` after every character of `s` and stores the resulting string in `w`. [3]

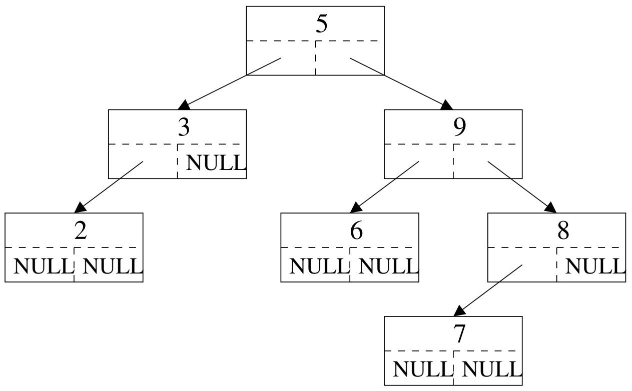
The question is about generic `w` with respect to `s` and `t`. Any specific answers for `s` and `t` being “abc” and “de” or any assumptions about the lengths of s and t being 3 and 2 respectively are incomplete.

[7] 6. (a) Consider a structure (or a node) as having three members (or fields). One is a value, and the other two are pointers to the same type of structure. The pointers can be named as `left` and `right`.

Such nodes can be used to create a *binary tree.* A binary tree is a collection of such nodes in a way that every node has at most two valid pointers. A valid pointer signifies that it is pointing to another node of the binary tree. Otherwise the value of the pointer is `NULL`.

One of the nodes of the tree is not pointed to by any other node. This node is called the *root* node (e.g., the one with the value 5 in the figure below), and we can draw the entire collection of nodes as an inverted tree. The node pointed to by the `left` and `right` pointers of the root node can be considered as root nodes of the corresponding sub-trees.

An example binary tree is given below:



We need to read values of all the nodes of a tree. These values can be read in different orders. A particular order can be recursively defined in three steps as follows:

1. Read values of the left sub-tree (i.e., all those nodes that can be reached through the `left` pointer of the root node)
2. Read the value of the root node
3. Read values of the right sub-tree (i.e., all those nodes that can be reached through the `right` pointer of the root node)

The procedure for reading the values in each of the sub-trees is the same and is done recursively. Note that the recursion ends when the pointer is `NULL`.

Using this algorithm, list the values of the nodes in the binary tree shown above in the order that this algorithm will read them in.

[6] (b) Assume an array of size $n+1$ that contains all the integers from 1 to n with one of them occurring twice. Describe an algorithm to detect the integer that is repeated. (You do not have to write the code for it.)

You get higher credit for a more efficient algorithm.

Answer:

(a) 2 3 5 6 9 7 8

A few sequences which could be achieved with minor errors in implementation would get you 3 or 4 marks. They are:

8 7 9 6 5 3 2	4 marks
5 3 2 9 6 8 7	4 marks
2 3 6 7 8 9 5	4 marks

Any two numbers interchanged from correct answer would also get you 4 marks.

3 2 5 9 6 8 7	3 marks
2 3 5 6 7 8 9	3 marks

Any sequence of 7 numbers, other than those mentioned above would get you 2 marks.
Any other sequence (either some number missing, or some number appearing twice or more) would give you 0 marks.
If you did not write the answer in sequence and drew a tree or any non-linear shape, then you get 0 marks.
(b)
The most efficient solution (6 marks) is:
Sum all n+1 elements of the array. From this sum subtract $n(n+1)/2$.

The second efficient solution (5 marks) is:
Have another array count[n+1]. Assume, original array is a[n+1]. In a for loop with i from 0 to n, count[a[i]]++. If this number becomes two, break, and print a[i].

The brute force method (2 marks) is:
for i = 0 to n, for j = 0 to n
if i != j and a[i] == a[j], break and print a[i]

Any variant of these methods which keep the processing time same, would get the same marks.
If the solution is not explained well (and that is indeed the problem with several students), then marks will be deducted.
Just declaring a count array does not mean that you have a variant of second solution. There are many who are updating the count in a nested "for" loop, and hence this will be considered a variant of third answer.
Many people have just stated, "first sort the array" without explaining the algorithm for sorting. If everything else is correct and efficient (that is, you just check for consecutive numbers to be same), then you get 1 mark, otherwise 0 marks.
Many people in variation of 2nd answer, have said that somehow the count array will count the number of times an integer is present in the original array. If you didn't say how the counting will happen, you have got a maximum of 2 marks.
If you have started declaring pointers, structures, trees, linked lists, etc., you get 0 marks.

[14] 7. The following function changes the size of the memory block pointed to by `p` which is equivalent to `n` integers to that equivalent to `m` integers. It returns another pointer that

points to the new memory block, and de-allocates the space held by `p`.

The content of the memory block is preserved up to the lesser of the new and old sizes. If the new size `m` is larger, the values of the rest of the elements are indeterminate or garbage.

When `p` is `NULL`, the function behaves exactly as `malloc`, i.e., it assigns a new block of memory sufficient for `m` integers and returns a pointer to the beginning of it.

When `m` is 0, a `NULL` pointer is returned.

Complete the code for the function `realloc`.

```
??1?? realloc(int *p, int n, int m)
{
  int *q;
  int lesser, i;
  if (m == 0)
  {
    ??2??
    return NULL;
  }
  q = (??3??)malloc(??4??);
  if (n < m)
    lesser = ??5??;
  else
    lesser = m;
  if (p == NULL)
    lesser = 0;
  for (??6??)  // copy the contents
  {
    ??7??
  }
  if (p != NULL)
    free(p);
  return ??8??;
}
```

Answer:

??1??: `int *` [2]
??2??: `free(p);` [1]
??3??: `int *` [1]
??4??: `m * sizeof(int)` [2, 1 for only `m` or wrong `sizeof`]
??5??: `n` [1]
??6??: `i = 0; i < lesser; i++` [3, 1 for each part]
??7??: `*(q + i) = *(p + i);` [3]

`*q++ = *p++` is also right, but then next answer must be `q - lesser`
??8??: `q` [1]

[2] 8. (a) Assume a symmetric matrix of size $n \times n$. Instead of storing it in a two-dimensional array having n rows and n columns, requiring n^2 amount of space, it can be stored more succinctly by just storing the lower triangular part of it. Any element that is in the upper triangular region can be deduced by the corresponding element in the lower triangular part.

The elements of the lower triangular part can be stored in a one-dimensional array in the following way. First, only the first row is stored. Then, the elements of the second row are stored. Next, the third row is stored and so on. Note that when a row is stored, only the elements that fall in the lower triangular part are stored.

What is the size required for this one-dimensional array? Explain.

[10] (b) Using the above way of storing a symmetric matrix, complete the following code segment that stores a matrix `m` of size $n \times n$ into an array `a` of appropriate size.

```
int k = 0, i, j;
for (??1??)
{
  for (??2??)
  {
    ??3??
  }
  ??4??
}
```

[3] (c) Consider a two-dimensional matrix of size 5x5 where each element of the matrix is either 0 or 1. Moreover, all the rows in the array are of the form 0's followed by 1's, i.e., the first part of the row consists of 0's and then the rest contains only 1's. Of course, in a particular row, there may be only 0's or only 1's.

This matrix can be alternatively stored in the form of a single dimensional array A of size 5 where A[i] stores the index of the first 1 in the ith row of the matrix. If there is no 1 in the row, A[i] stores 5.

For example, if the matrix is
1 1 1 1 1
0 0 1 1 1
0 1 1 1 1
0 0 1 1 1
0 0 0 0 0
then the corresponding single dimensional array A is {0, 2, 1, 2, 5}.

Complete the following function `convert` that takes this single dimension array A and two indices i and j as input and returns the (i,j)th entry of the original two-dimensional matrix.

```
int convert(int A[], int i, int j)
{
  ??1??
  return 0;
  ??2??
  return 1;
}
```

Answer:

(a) $n(n+1)/2$ as that is the size of the lower triangular part $(1 + 2 + \cdots n)$.
(b)
??1??: `i = 0; i < n; i++` [2]
??2??: `j = 0; j <= i; j++` [3]
??3??: `a[k++] = m[i][j];` [4]
??4??: [1]
(c)
??1??: `if (j < A[i])` [2]
??2??: `else` or `if (j >= A[i])` [1]