CS10001   Programming and Data Structures

End-semester Examination
Autumn 2008

Duration: 3 hrs **Total Marks: 100**

Name:	
Roll No:	
Section:	

Answer in the question paper itself. Do your rough work on reverse side of the question sheet. Do not leave the examination hall without submitting the question paper. Write your roll no on top of every page of the question paper. There are 12 pages in the question paper.

Answer Question 1, and ANY FOUR from the rest.

Question 1 (20)	2 (20)	3 (20)	4 (20)	5 (20)	6 (20)	Total (100)

Roll No:

1.1. Suppose *z* is a k-bit binary number. Then the binary of $2^k - z$ represents: **[2]**

a. 1's compliment of *z*
b. 2's compliment of *z*
c. None of these

(b) 2's compliment of z

1.2. Consider the number 1110 in a 4-bit 2's compliment representation. Rewrite the same number in a 8-bit 2's compliment representation. **[2]**

11111110

1.3. Suppose we have the following statement inside a function `foo( )`, and `int *p` is a global variable: **[2]**

```
p = (int *) malloc(50 * sizeof(int));
```

Suppose `main( )` calls `foo( )`. From which of the following places can we access the memory allocated by this call?

a. Only within the function `foo( )`
b. Within the function `foo( )` and the function `main( )`
c. Within any function, that is, the memory is global

(c) Within any function, that is, the memory is global

1.4. What is the data type of `A`, when it is declared as `int *A[ ]` ? **[2]**

a. An array of integers
b. A pointer to an array of integers
c. An array of pointers to integers

(c) An array of pointers to integers

1.5. Convert the decimal number **371** to binary and hexadecimal: **[2]**

Binary: **0101110011**

Hex : **173**

1.6. Convert the decimal number **4.75** to binary and hexadecimal: **[2]**

Binary: **0100.110**

Hex: **4.C**

Roll No:

1.7. What value is returned by the call `f(1)` ? **[2]**

```
int f (int n)
{
      static int i = 1;
      if (n >= 5) return n;
      n = n + i;
      i++;
      return f(n);
}
```

7

1.8. Write the values of the following expressions **[2]**

a. ‘F’ – ‘C’

b. 2.0 + (float) (5/3)

(a) **3**

(b) **3.0**

1.9. How many *'s are printed by the following code segment ? **[2]**

```
for (i=0; i< 10; i++)
      for (j=i; j>0; j--)
            printf(“*”) ;
```

45

1.10. How many *s will be printed by the following program ? **[2]**

```
#include <stdio.h>
int F[10] ;

int fib( int n )
{
      printf(“*”);
      if (n <= 1) return 1;
      if (F[n] != 0) return F[n];
      F[n]=(fib(n-1) + fib(n-2));
      return F[n] ;
}

int main( )
{
      int j;
      for (j=0; j< 10; j++) F[j] = 0;
      fib(5);
}
```

9

Roll No:

2. The absolute distance between two integers *x1* and *x2* is given by | *x2 – x1* |. Write a function which sorts an array *x[ ]* of n integers in ascending order of their absolute distances with a given number *z*. For example, given *x[ ] = {9, 1, 12, 4, 2}* and *z = 6*, the sorted array will be *x[ ] = {4, 9, 2, 1, 12}*. Note that 4 is closest to 6, and 12 is farthest from 6, in terms of absolute distances. The function will have the following prototype: **[20]**

```
void dist_sort( int x[ ], int n, int z ) ;
```

Roll No:

Roll No:

3. Write a void function, `void read_names(<arg1>, <arg2>)`, that does the following:

a. The function prompts the user to enter the number of students in a class and reads it into `arg2`
b. The function reads that many names and stores the names in the two dimensional array, `arg1`. The two dimensional array is dynamically allocated within the function such that the number of characters allocated in the i^{th} row is equal to the length of the i^{th} name plus one (for the `\0`).

Finally, write a `main( )` which calls the function `read_names.` **[20]**

Roll No:

Roll No:

4.1 Define a structure ‘`pt`’ to represent a point in two dimensions. Use the `pt` structure to define a structure ‘`rect`’ representing a rectangle (in two dimensions). **[2 + 3]**

Point

Rectangle

4.2 Write a function which takes as input an integer *n*, and an array of *n* rectangles, and returns the area of the smallest rectangle enclosing all the *n* rectangles in the array. The function prototype is as follows:

```
int bounding_rectangle ( int n, struct rect rect_array[ ] );
```

[15]

Roll No:

Roll No:

5. A polynomial $A(x)= a_0 + a_1x + a_2x^2 + .. + a_nx^n$ may be written as $A(x) = a_0 + x(a_1 + x(a_2 +.. x(a_n)))$.

Using this rule (called the Horner's rule) a polynomial may be evaluated at a point *x*, that is *A(x)* can be computed, by repeated multiplications and additions, rather than the naive methods of raising *x* to powers, multiplying by the coefficient, and adding.

Write a function which takes as input : (a) an integer *n* (degree of the polynomial), (b) a float array *A[ ]* containing the coefficients of the polynomial (a_0, a_1, a_2..a_n), and (c) another float *x*. The function should return the value of the polynomial evaluated at *x*, *A(x)* using the Horner's rule mentioned above. The prototype is as follows:

```
float polynomial(int n, float A[ ], float x) ;
```

[20]

Roll No:

6. Consider the type definitions given below. Suppose that a linked list is made up of nodes of type `struct node`. The last node points to NULL.

```
struct node {
        int key;
        struct node * next;
     };
     typedef struct node* link;
```

6.1. Write a function `deletesecond()` to delete the second node of the list, given "*head*" which points to the first element of the list. Assume that there are at least two nodes on the list. **[5]**

```
void deletesecond (link head)  {










}
```

6.2. Write a function `count()` that takes the head of a linked list as input, and returns the number of nodes in the linked list. **[5]**

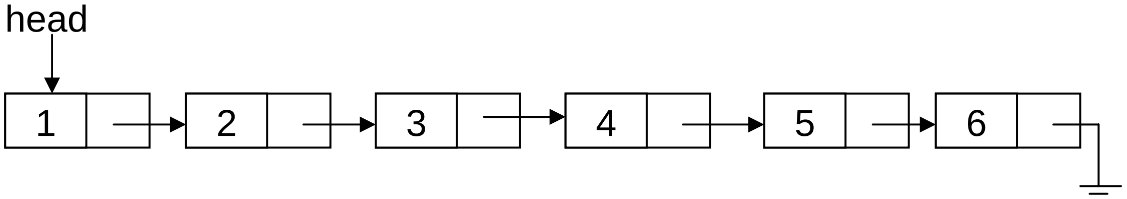
```
int count (link head)  {










}
```

Roll No:

6.3. Consider the following function `baz()` that takes a pointer to the first node of a linked list.

```
void baz (link head)  {
      if (head == NULL)
             return ;
      baz (head->next->next) ;
      printf (“%d ”, head->key) ;
}
```

What will be printed when the function is called with the linked list shown below? **[5]**



6.4. Consider the same linked list as shown in the previous figure, with “head” pointing to the first node of the list. Show the changes in the list structure after the following code segment is executed by drawing a diagram. Clearly indicate where, `q` and `p` point to after the execution of the program segment. **[5]**

```
link head, q, p;
   q = NULL;
   while(head!=NULL)    {
          p=head;
          head = head->next;
          p->next = q;
          q = p;
   }
```