INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Date: 28/04/2011 *FN* Time: *3 hrs* Full marks: 100 No. of students: 643

Spring End Semester Exams, 2010-11 Dept: *Comp. Sc & Engg.* Sub No: ***CS11001***

B.Tech 1st Year (Core) Sub Name: ***Programming and Data Structures***

Instructions:

ANSWER Q.1 AND ANY FOUR OF THE REMAINING FIVE QUESTIONS

Write answers in the space provided in the question paper itself.

Do your rough work on the space provided on the back sides of the printed sheets.

Write your roll number at the space provided in every sheet of the question paper.

Name:
Roll No:
Section:

-------------------------- DO NOT WRITE ANYTHING BELOW THIS LINE ------------------------------

MARKS

Q1 (20)	Q2 (20)	Q3 (20)	Q4 (20)	Q5 (20)	Q6 (20)	Total (100)

ROLL NUMBER: ________________

1. Compulsory question. Answer all parts of this question.

(a) The array `int a[10][10]` is filled with an $r \times c$ ($0 \leq r, c < 10$) matrix where `a[i][j] = 10*i+j`. What will be printed due to the function call:

```
what ( (int (*)[10])&a[2][2], 3, 3);
```

where the function definition is as follows.

```
void what(int x[][10], int r, int c){
  int i, j;
  for(i=0; i > -r; --i){
     for(j=0; j > -c; --j) printf("%d ", x[i][j]);
     putchar('\n');
  }
}
```

[5 marks]

(b) Suppose that *i* and *j* are both of type `int`. What is the value of *j* after each of the following statements is executed?

```
for (i=0, j=1; i < 10; i++) j+= j;
for (j=0; j < 10; j++) j+= j;
```

[3 marks]

Value of *j* after the first *for* statement =

Value of *j* after the second *for* statement =

(c) What does the following program print on input: `IIT 4.2 5`

```
#include <stdio.h>
int main()
{
   char a[20], b;
   int x;
   float y;

   scanf("%s%c%d%f", a, &b, &x, &y);
   printf("a=%s, b=%c, x=%d, y=%2.1f\n", a, b, x, y);
   return 0;
}
```

[2 marks]

(d) Study the following function and count the number of times `foo(4)` is called within a call to `foo(10)`.

```
float foo(int n)
{
      float x, y, z;
      scanf("%f%f", &x, &y);
      z= (x*foo(n-1))+(y*foo(n-1));
      return z;
}
```

[3 marks]

(e) Which of the following type definitions and declarations is / are correct?

Option-1	Option-2	Option-3

`typedef struct t{`
`int key;`
`struct t *next;`
`} node;`
`node z;`
	`typedef struct {`
	`int key;`
	`node *next;`
	`} node;`
	`node z;`
		`typedef struct node {`
		`int key;`
		`struct node *next;`
		`};`
		`node z;`

[2 marks]

(f) What does the following program print?

```
#include <stdio.h>
#include <stdlib.h>
struct node { int key; struct node * next; };

int main()
{
    struct node *x, *y, *t;
    x = (struct node *) malloc(sizeof(struct node));
    y = (struct node *) malloc(sizeof(struct node));
    x->next = y; x->key = 1;
    y->next = x; y->key = 1;
    for (t=x; t->key < 100; t=t->next)
        t->key = x->key + y->key;
    printf("%d\n", t->key);
    return 0;
}
```

[5 marks]

ROLL NUMBER: ________________

2. Answer the following questions. [10 + 10 = 20 marks]

(a) The following figure shows a 4×4 square matrix, *X*, and its *rotation*, *Y*.

$$X = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} \qquad Y = \begin{bmatrix} 13 & 9 & 5 & 1 \\ 14 & 10 & 6 & 2 \\ 15 & 11 & 7 & 3 \\ 16 & 12 & 8 & 4 \end{bmatrix}$$

Complete the function **`rotate( )`** in the following program to rotate a *N*×*N* matrix *X* and store the rotated matrix in *Y*.

```
#define N 4
#include <stdio.h>

void rotate(                          )
{










}

int main()
{
    int X[][N] = {{1, 2, 3, 4}, {5, 6, 7, 8},
    {9, 10, 11, 12}, {13, 14, 15, 16}}, Y[N][N];

    rotate(X, Y);
}
```

ROLL NUMBER: ________________

3. *n* integer data items are stored in a 1-D array. A pair of data **a[i]** and **a[j]** is called an ***inversion pair*** if *i* < *j* and **a[i] > a[j]**.
A *merge sort* function can be modified to return the count of *inversion pairs* along with sorting. The following **merge()** function takes four parameters; a pointer **x** to the beginning of the data array, low-index **l**, mid-index **m**, and the high-index **h**. The data from **x[l]** to **x[m]** and data from **x[m+1]** to **x[h]** are already sorted. The merge function *merges* them using the local array **t[MAXNO]** to get sorted data from **x[l]** to **x[h]** (where $0 \le l \le m \le h <$ MAXNO). The expression β_1 within the merge function is used to compute inversion pairs while merging two sorted parts.

```
int merge(int x[], int l, int m, int h){
   int t[MAXNO], i, j, k, invCount=0;

   for(i=l; i<= h; ++i) t[i] = x[i];
   i=l, j=m+1, k=l;
   while(i <= m && j <= h)
          if( t[i] <= t[j] )  x[k++] = t[i++] ;
          else {
                  β1;
                  x[k++] = t[j++];
          }
   if (i > m)
                for(; j <= h; ++j, ++k)  x[k] = t[j];
   else
                for(; i <= m; ++i, ++k)  x[k] = t[i];
   return invCount;
}
```

[6 + 14 = 20 marks]

(a) What is the expression for β_1.

β_1:

(b) Write the function **int mergeSort(int x[], int low, int high)** that uses the given function **merge()**, sorts the data and also returns the total count of *inversion pairs* in the input data. The input data is present in the array pointed by **x[]** and from the index **low** to the index **high**.

ROLL NUMBER: ________________

Ans 4(a):

ROLL NUMBER: ________________

(b) Convert the decimal number 1.23 to binary. Show the steps of the computation.

Ans 4(b):

5. Consider the following inductive definition (recurrence relation) of a function on non-negative integers.

```
f(n)  = n                     if n < 3
      = 2f(n-2) + 3f(n-3)     otherwise
```

A few values are:

n	0	1	2	3	4	5	6	…
f(n)	0	1	2	2	7	10	20	…

[6 + 7 + 7 = 20 marks]

ROLL NUMBER: ________________

(a) Write a recursive function **`int funR(int n)`** that directly encodes the definition to computes and returns the value of *f(n)* when *n* is passed as the parameter.

Ans 5(a):

(b) Write an iterative function **`int funI(int n)`** that computes and returns the value of *f(n)* when *n* is passed as the parameter.

Ans 5(b):

ROLL NUMBER: ________________

(c) The following recursive function (incomplete) computes `f(n)` efficiently.

```
int funER (int n, int f0, int f1, int f2)  {
      if(n < 2) return β1;
      if (n == 2) return β2;
      return funER (β3, f1, β4, β5);
 }
```

Fill-in β_1, β_2, β_3, β_4 and β_5 with appropriate expressions.
How will the function be invoked to compute **f(15)**?

ANS 5(c):

β_1:____

β_2:____

β_3:____

β_4:____

β_5:____

To compute **f(15)**, the function will be invoked as:

____

6. Consider the problem of representing a set (of integers) as a linked list. As per the definition of a set, it will consist of unique elements. The following structure definition is used to represent a node of the linked list:

```
struct setnode  {
      int  member;
      struct  setnode  *next;
};

typedef  struct  setnode  SET;
```

A set, represented by the linked list as mentioned above, can be accessed through a pointer pointing to the first node of the list.

(a) Write a C function to insert an element in a set *at the end of the list* provided it is not already in the set. The function will return a pointer to the first node of the set. The function prototype will be:

```
SET  *insert_element  (SET *myset, int element);
```

which will insert **element** into the set pointed to by **myset**.

(b) Write a C function to create a set. The function will read **n** integers, where **n** is a parameter, and then insert them one-by-one into the set using the function **insert_element()**. It will return a pointer to the first node of the set. The function prototype will be:

```
SET  *read_set (int n);
```

(c) Write a C function to compute and return the intersection of two sets, with the following prototype:

```
SET  *intersect  (SET *setA,  SET *setB);
```

setA and **setB** are pointers to the first elements of the two given sets whose intersection is to be found out.

[8+4+8=20 Marks]

ROLL NUMBER: ________________

Ans 6(a):

Ans 6(b):

ROLL NUMBER: ________________

Ans 6(c):