INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

27 NOV 2018

Stamp / Signature of the Invigilator

EXAMINATION (End Semester)										SEMESTER (Autumn)	
Roll Number									Section		Name
Subject Number	C	S	1	1	0	0	1	Subject Name		*Programming and Data Structures*	
Department / Center of the Student										Additional sheets	

Instructions and Guidelines to Students Appearing in the Examination

1. Ensure that you have occupied the seat as per the examination schedule.
2. Ensure that you do not have a mobile phone or a similar gadget with you even in switched off mode. Note that loose papers, notes, books should not be in your possession, even if those are irrelevant to the paper you are writing.
3. Data book, codes or any other materials are allowed only under the instruction of the paper-setter.
4. Use of instrument box, pencil box and non-programmable calculator is allowed during the examination. However, exchange of these items is not permitted.
5. Additional sheets, graph papers and relevant tables will be provided on request.
6. Write on both sides of the answer script and do not tear off any page. Report to the invigilator if the answer script has torn page(s).
7. Show the admit card / identity card whenever asked for by the invigilator. It is your responsibility to ensure that your attendance is recorded by the invigilator.
8. You may leave the examination hall for wash room or for drinking water, but not before one hour after the commencement of the examination. Record your absence from the examination hall in the register provided. Smoking and consumption of any kind of beverages is not allowed inside the examination hall.
9. After the completion of the examination, do not leave the seat until the invigilator collects the answer script.
10. During the examination, either inside the examination hall or outside the examination hall, gathering information from any kind of sources or any such attempts, exchange or helping in exchange of information with others or any such attempts will be treated as adopting 'unfair means'. Do not adopt 'unfair means' and do not indulge in unseemly behavior as well.

Violation of any of the instructions may lead to disciplinary action.

Signature of the Student

To be filled in by the examiner

Question Number	1	2	3	4	5	6	7	8	9	10	Total
Marks Obtained											

Marks obtained (in words)	Signature of the Examiner	Signature of the Scrutineer

Instructions: Answer all seven questions. Total marks = 15 x 2 + 14 x 5 = 100. Time = 3hrs. Write your answer only in the space provided. Use any other space for rough work. The question paper has total 12 pages. **WRITE YOUR SECTION NUMBER IN THE FIRST PAGE.**

Rough Work

1. (i) The base (or radix) of the number system such that the equation 312/20 = 13.1 holds, is?

(A) 3 (B) 4 (C) 5 (D) 6

(ii) Consider the following C function in which *size* is the number of elements in the array *E*: The value returned by the function *MyX* is the

```
int MyX(int *E, int size)
{
    int Y = 0; int Z; int i, j, k;
    for(i = 0; i < size; i++) Y = Y + E[i];
    for(i = 0; i < size; i++)
        for(j = i; j < size; j++)
        {
            Z = 0;
            for(k = i; k <= j; k++)
                Z = Z + E[k];
            if (Z > Y)
                Y = Z;
        }
    return Y;
}
```

(A) maximum possible sum of elements in any subarray of array E.

(B) maximum element in any sub-array of array E.

(C) sum of the maximum elements in all possible sub-arrays of array E

(D) the sum of all the elements in the array E.

(iii) What is the output of the following c code? Assume that the address of *x* is 2000 (in decimal) and an integer requires four bytes of memory.

```
#include <stdio.h>
main()   {
   unsigned int x[4][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}, {10, 11, 12}};
   printf("%u, %u, %u", **x+3, **(x+3), **(x+2)+3);     }
```

(A) 4 10 10 (B) 4 4 4 (C) 4 10 6 (D) 4 4 6

(iv) What is the output of the following program fragment?

```
int a=2,*f1;
f1=&a; *f1 += (a += 2);
printf("a= %d *f1= %d", a, *f1);
```

(A) 4 4 (B) 4 6 (C) 6 6 (D) 6 4

(v) Consider the following c code. After the following lines of code, which of the following lines will change the value of *i* to 75? `int *p; int i; int k; i = 42; k = i; p = &i;`

(A) `k = 75;` (B) `*k = 75;` (C) `p = 75;` (D) `*p = 75;`

(vi) The following c function takes a single-linked list of integers as a parameter and rearranges the elements of the list. The function is called with the list containing the integers 1,2,3,4,5,6,7 in the given order. What will be the contents of the list after the function completes execution?

```
struct node { int value; struct node * next; };

void rearrange struct node * list {
struct node *p, *q;
int temp;
if (!list || !list -> next) return;
p = list; q = list -> next;
while (q) {
     temp = p -> value; p -> value = q -> value;
     q ->value =temp; p= q ->next;
     q = p ? p - > next : 0;
}   }
```

(A) 1,2,3, 4,5, 6,7 (B) 2,1, 4,3, 6,5,7 (C) 1,3,2,5, 4,7, 6 (D) 2,3, 4,5, 6,7,1

(vii) If the following sequence of operations is performed on a stack, the sequence of popped out values is *push(1); push(2); pop; push(1); push(2); pop; pop; pop; push(2); pop;*

(A) 2 2 1 1 2 (B) 2 2 1 2 2 (C) 2 1 2 2 1 (D) 2 1 2 2 2

(viii) Consider the following operations on a Queue data structure that stores int values. What value is returned by the last dequeue operation?

enqueue(3); enqueue(5); enqueue(9); dequeue( ); enqueue(2); enqueue(4); dequeue( );dequeue( );

(A) 3 (B) 9 (C) 5 (D) 2

(ix) Suppose we have a circular array implementation of the queue, with ten items in the queue stored at data[2] through data[11]. The current capacity is 12. Where does the insert method place the new entry in the array?

(A) data[1] (B) data[0] (C) data[11] (D) data[12]

(x) Consider the following c function. What is the value of *f(5)*?

```
int f(int n)    {
static int r = 0;
if (n <= 0) return 1;
if (n > 3)
     {r = n;
     return f(n-2)+2;
     }
return f(n-1)+r;    }
```

(A) 19 (B) 20 (C) 16 (D) 18

(xi) What will be the output of the following code-segment?

```
int i, num[5]; int *j;
for (i=0; i < 5; i++) num[i] = i * 5;
j = num;
for(i=0; i<5; i++) printf("%d\n", (*j)++);
```

(A) 0 5 10 15 20 (B) 1 6 11 16 21 (C) 1 2 3 4 5 (D) 0 1 2 3 4

(xii) What will be the output of the following code-segment?

```
int a[] = {1, 7, 9, 5, 6 , 8, 9, 7, 4, 9};
int *p = a +3;
int *q = a + 6;
printf ("\n %ld", q-p);
```

(A) 3 (B) 4 (C) -1 (D) 0

(xiii) Consider the following c function. What is printed after the call *f(3)*?

```
void fun(int p)
{
  if(p > 0)
   {
     fun(--p);
     printf("%d", p);
   }
}
```

(A) 0 1 2 3 (B) 1 2 3 0 (C) 3 2 1 9 (D) 1 2 3 4

(xiv) What is the value of *func(5)* ?

```
int func (int x)
 {
   if (x <= 0)
   return(1);
   return func(x - 1) + 2*x;
 }
```

(A) 30 (B) 31 (C) 32 (D) 33

(xv) The number of comparisons required in Binary Search for an array of n elements is

(A) n (B) $n/2$ (C) $\log_2 n$ (D) $n.\log_2 n$

2. Let *ABCD* be a parallelogram. The co-ordinates of the corners $A\ (x_A, y_A)$, $B\ (x_B, y_B)$, $C\ (x_C, y_C)$ are input by user. Assume *A, B, C* to be non-collinear. Complete the following program to print the co-ordinates of the corner $D\ (x_D, y_D)$. Hint: *D* is the intersection of the lines *AD* and *CD*. **[14]**

```
int main(){
float xa, ya, xb, yb, xc, yc, xd, yd;
float mba, mbc, cad, ccd;
scanf("%f %f %f %f %f %f", &xa, &ya, &xb, &yb, &xc, &yc);

if ((xa != xb)&& (xc != xd)) {

      mba = _____; // slope of BA

      mbc = _____; // slope of BC

      cad = mbc*xa - ya;          // y-axis intercept of AD
      ccd = mba*xc - yc;          // y-axis intercept of CD

      xd = _____;

      yd = mbc*xd + cad;       }

else if (xa == xb)   {

      xd =_____;

      yd =_____; }

else if (xc == xb)   {

      xd = _____;

      yd = _____; }

      printf("xd = %f yd = %f\n", xd, yd);

return 0;}
```

3. An atomic element can be defined as a structure containing a char representing its symbol, atomic number of the element, and its atomic weight. A compound can be represented by a structure containing the number of elements present in the compound *n*, and array, *atoms*, of elements, and another array, *atmcnt*, storing the count of the number of atoms of each element in the compound.

i. Complete the function *molecularwt()*, which takes as input a molecule and returns its molecular weight.

```
struct element {char symbol; int atomicno; float atomicwt;};
struct compound {int n; struct element atoms[10]; int atmcnt[10];};

float molecularwt(struct compound r){
int i; float wt = 0;
for (i=0; i < _____ ; i++)

      wt += _____;

return wt;   }
```

ii. Consider a chemical equation *r1 + r2 = r3*, of two reactant molecules (*r1* and *r2*) reacting to produce a product molecule (*r3*). The molecules *r1* and *r2* do not have any element in common. Complete the function *isbalanaced( )*, which takes as input *r1, r2, r3,* and checks if the equation is balanced in terms of the number of atoms of each element involved. For example, C6 + H12O6 = C6H12O6, is a balanced equation. **[4 + 10]**

```
int isbalanced(struct compound r1, struct compound r2, struct compound r3){
int symb[20], elcnt[20], i, j, flag, aflag=0;

for(i=0; i < r1.n; i++){
      symb[i]= r1.atoms[i].symbol;
      elcnt[i] = r1.atmcnt[i];  }

for(i=0; i < r2.n; i++){
      symb[i+r1.n]= r2.atoms[i].symbol;
      elcnt[i+r1.n] = r2.atmcnt[i];  }

for(i=0;i < r3.n; i++){
      flag = 1;
      for(j=0; j< _____; j++){

            if (symb[j] == _____)

                  if(elcnt[j] == _____)

                        flag = _____; }
aflag = aflag + flag; }

if(_____) return 1;
else return 0;
}
```

4.i. Complete the function *spiralmatrix( )* which takes as input an integer n, and returns a $n \times n$ matrix containing integers $1, \ldots, n^2$ in a clock-wise spiral arrangement as shown below.

1	**2**	**3**	**4**
12	**13**	**14**	**5**
11	**16**	**15**	**6**
10	**9**	**8**	**7**

```
void spiralmatrix(int n, int a[][50])  {
      int value=1,i=0,j=0,k,m;
      m=n;

      while(n>0)  {
           a[i][j++]=value++;

           for(k=1;k<=n-1;k++)  _____________________=  value++;

           i++; j--;

           for(k=1;k<=n-1;k++)  _____________________=  value++;

           i--; j--;

           for(k=1;k<=n-1;k++)  _____________________=  value++;

           i--; j++;

           for(k=1;k<=n-2;k++)  _____________________=  value++;

           i++;  j++;   n = n-2;
      } }
```

ii. Complete the function *manhattandist( )*, which takes as input an integer n, a 2-D array *A[ ][ ]* of size $n \times n$ storing integers $1, \ldots, n^2$ in spiral arrangement as mentioned above, and two integers x_1 and x_2. The function returns the Manhattan distance between the array elements storing x_1 and x_2. The Manhattan distance between the array elements with indices *(i1, j1)* and *(i2, j2)* is defined as $|i1 - i2| + |j1 - j2|$. **[8+6 = 14]**

```
int manhattandist(int n, int A[][50], int x1, int x2) {
     int i,j,i1,j1,i2,j2,dist=0;
     for(i=0;i<=n-1;i++)
     {
         for(j=0;j<=n-1;j++)
            {
              if(x1==A[i][j]) {i1=i;j1=j;}
              if(x2==A[i][j]) {i2=i;j2=j;}
            }
     }
     if(i1 >= i2) dist = _____________________________ ;

     else dist = _____________________________ ;

     if(j1 >= j2) dist = _____________________________ ;

     else dist = _____________________________ ;

     return(dist);
}
```

5. (a) Write a suitable *typedef* to represent a linked list of integers.

```
typedef struct ll_node
{
  _____________________________;

  _____________________________;
} node;
```

(b) Write a c function *RemoveDuplicates()* which takes a list sorted in increasing order and deletes any duplicate nodes from the list. Ideally, the list should only be traversed once.

```
void RemoveDuplicates(node* head) // Remove duplcats in a sorted list
{
  node* current = head, *nextNext;

  if (current == NULL) return;     // do nothing if the list is empty

  while(current->next != NULL)
    { // Compare current node with next node

      if (_____________________________________________)
      {
        _____________________________________________;
        free(current->next);

        current->next = _____________________________;
      }
      else
      {
        current = _____________;  // only advance if no deletion
      }
    }  }
```

(c) In a circular linked list, the next pointer of the last node points to the starting node of the list. Write a recursive c function that prints the elements of a circular linked list of integers in the reverse order (that is, from end to beginning). Use the function prototype: *void printCircList (circlist l, const circlist h);* Here the second parameter points to the beginning of the list and is kept constant across the calls. Assume that no dummy header node is used in the circular linked list. **[2+8+4 = 14]**

```
void printCList ( node* l, const node* h )  {

if(_____________________________) {
      printf("%d ", l->data);
      return;    }

printCList(_____________________);
printf("%d ", l->data);

if(l == h) printf("\n");
}
```

6. Consider a sorted array *X* of *n* elements. Given a *key* to be searched, the array is partitioned into three halves at the elements *mid1* and *mid2*. *mid1* indicates 1/3rd index and *mid2* indicates 2/3rd index of the sorted array. Now comparing the key value to *mid1* or *mid2* it is determined whether the key lies in first third, middle third or last third of the array. Continue the process until the key is found, if it exists in the array. This method of searching is called *Ternary Search*. Complete the function *TernarySearch()*, in recursive as well as non-recursive way, which takes as input (i) the array, (ii) left and (iii) right index of the sorted array and (iv) the key to be searched and returns the value of key, if it exists in the array otherwise returns -1. **[9 + 5 = 14]**

```
// Recursive Function
int TernarySearch(int x[], int left, int right, int key) {
  int mid1,mid2;
  if(left<=right)
   {
     mid1=left+(right-left)/3;  mid2=left+(right-left)*2/3;
     if(key==x[mid1]) return(mid1);
     else if(key<x[mid1])

       return(_____);

     else if(key==x[mid2]) return(mid2);
     else if(key<x[mid2])

       return(_____);

     else

       return(_____);
   }
   else return(-1);  }          // KEY IS NOT FOUND
```

```
// Non-Recursive Function
int TernarySearch(int x[], int left, int right, int key) {
    int mid1,mid2;
    while(left<=right)
      {
        mid1=left+(right-left)/3;
        mid2=left+(right-left)*2/3;
        if(key == x[mid1]) return(mid1);
        else
          if(key<x[mid1])  {_____}
          else
            if(key == x[mid2]) return(mid2);
            else
             if(key<x[mid2])
             {
                 _____;

                 _____;
             }
            else left = mid2+1;
   }
    return(-1);  }                // KEY IS NOT FOU
```