INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR												**Stamp / Signature of the Invigilator**
EXAMINATION (Mid Semester)										**SEMESTER (Autumn)**		
Roll Number									**Section**		**Name**	
Subject Number	C	S						**Subject Name**		*Programming and Data Structures*		
Department / Center of the Student										**Additional sheets**		

Important Instructions and Guidelines for Students

1. You must occupy your seat as per the Examination Schedule/Sitting Plan.
2. Do not keep mobile phones or any similar electronic gadgets with you even in the switched off mode.
3. Loose papers, class notes, books or any such materials must not be in your possession, even if they are irrelevant to the subject you are taking examination.
4. Data book, codes, graph papers, relevant standard tables/charts or any other materials are allowed only when instructed by the paper-setter.
5. Use of instrument box, pencil box and non-programmable calculator is allowed during the examination. However, exchange of these items or any other papers (including question papers) is not permitted.
6. Write on both sides of the answer script and do not tear off any page. **Use last page(s) of the answer script for rough work.** Report to the invigilator if the answer script has torn or distorted page(s).
7. It is your responsibility to ensure that you have signed the Attendance Sheet. Keep your Admit Card/Identity Card on the desk for checking by the invigilator.
8. You may leave the examination hall for wash room or for drinking water for a very short period. Record your absence from the Examination Hall in the register provided. Smoking and the consumption of any kind of beverages are strictly prohibited inside the Examination Hall.
9. Do not leave the Examination Hall without submitting your answer script to the invigilator. **In any case, you are not allowed to take away the answer script with you.** After the completion of the examination, do not leave the seat until the invigilators collect all the answer scripts.
10. During the examination, either inside or outside the Examination Hall, gathering information from any kind of sources or exchanging information with others or any such attempt will be treated as **'unfair means'**. Do not adopt unfair means and do not indulge in unseemly behavior.

Violation of any of the above instructions may lead to severe punishment.

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**

1. (9 marks) Write C statements (program segments only) of a program that reads the lengths of the sides of a triangle to find the nature of the triangle. You are required to only write program segments for the following tasks only and not the complete program. **Marks:** $1+1+2+2+3$

(a) Declare variables `a`, `b` and `c` of type `float`.

Solution: `float a,b,c;` No part marks

(b) Read `a, b, c`.

Solution: `scanf("%f %f %f",&a,&b,&c);` No part marks. %[a,e,g] are also correct in place of %f.

(c) Check if `a` contains the largest value (larger than `b` and `c`). If not, print an error message.

Solution:
```
if(a<b) printf("Error:  a not largest\n");
else if(a<c) printf("Error:  a not largest\n");
```
Other version of logic possible. 1 mark if only one of the conditions are checked.

(d) Write a program fragment to check and print whether `a, b, c` form the sides of a valid triangle. Assume that `a` has a value larger than `b` and `c`.

Solution: `if(b+c < a) printf("Error:  not a valid triangle\n");`

(e) Print "acute", "right-angled" or "obtuse", depending on the type of triangle formed by the sides `a, b, c`. Assume `a` is the largest side.

Solution:
```
if(b*b + c*c < a*a) printf("Obtuse triangle\n");
else if(b*b + c*c > a*a) printf("Acute triangle\n");
else printf("Right-angled triangle\n");
```

2. (7 marks) Complete the following C program so that it computes the sum of the following series upto n terms. **Marks:** $2+2+2+1$

$$1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\cdots$$

```
/* Compute the sum of the series  [ 1-X^2/2!+X^4/4!- ...] */
#include < stdio.h >
 int main()
 \{
       float x, sum, term;
       int i, n;

       printf("Enter the value of x and the number of terms to sum\n");
       scanf("%f%d", &x, &n);

       // Initialize values

       sum=1;__________________________________________________

       term=1;_________________________________________________


       for (i = 1; i < n; i++_________________________________)
       \{


             term = term *  ((-1*x*x)/(2*i)*(2*i-1));____________

                  _______________________________________________

             sum = sum + term;___________________________________

         printf("\n Sum = %f\n");
         return 1;
\}
```

3. (6 marks) Complete the following C program which given an input string prints whether it is a palindrome or not. **Marks:** $1+2+1+1+1$

```
#include <stdio.h >
int main() {
      char a[100];
      int i, j, length_a;

      printf("Enter the String(max length 100): ");
      // Read the string

      scanf("%s",a);____________________________________________);


      // Compute the length of string a using a loop and store it in
      // length_a. Do not use any library function.


       for(i=0;i<100;i++) __________________________________________


       ____if(a[i]=='\0') break;____________________________________


       length_a=i+1;________________________________________________


       _____________________________________________________________


      for (i = 0; i < length_a/2; i++) {


              if (a[i] == a[length_a-i-1]_______________________)
                      continue;
               else {

                      printf(_"Not a palindrome\n"______________________);
                      return 0;
              }

      }

      printf(_"String is a palindrome\n"______________________________);
      return 0;
 }
```

4. (5 marks) Write C program statements in the blanks such that the following function returns the minimum element in the array `a[ ]` between indices `start` and `end` (both inclusive): **Marks:** $1+1+3$

```
int minv_arr (int a[], int start, int end) {
    int temp;

    if( __start >= end________________________) // base condition

          return  ___a[start];________________________

    else {
        // Make the recursive call and return the minimum element.
        // You are not allowed to use any loop


            _____temp=minv_arr(a,start,end-1);____________________________

           _______if(temp<a[end])________________________________________

           ___________printf("Minimum is: %d\n",temp);________________________

          ________else printf(Minimum is: %d\n", a[end]);_____________________
    }
}
```

5. (10 marks) Write a program that takes as input `n`, followed by `n` integer numbers and store them in an array `A`. It then calls a function which copies the distinct elements of array `A` to an integer array `B` so that array `B` contains all elements of `A` but does not repeat any element. For example, if `A` stores $\{17,2,17,19,5,2,9,9,8,2\}$, array `B` will contain $\{17,2,19,5,9,8\}$ after the function call. The program comprises of a `main( )`, the function `makeset( )` and the function `check( )` which is called by `makeset( )`. **Marks:** $3+5+2$

The function `printarray( )` is given which takes as input an array of integers `A` and its length `n` and prints the array.

```
void printarray (int A[], int n) {
      int i;
      for (i = 0; i < n; i++) printf ("%d ", A[i]) ;
      printf("\n") ;
}
```

(a) Write the function `check( )` which takes as input an integer `x`, an array `A` and its size `n`. It should return `1` if `x` is occurs in array `A` and `0` otherwise

Solution:

```
int check(int x, int A[], int n) {
int i;
for(i=0;i<n;i++)
if(a[i]==x) return 1;
return 0;
}
```

(b) Write the function `makeset( )` which takes as input an array of integers `A`, its size `n1`, and an array of integers `B`, The function must copy the unique elements of `A` into the array `B` and return the number of elements in `B`, by making use of calls to the function `check( )` defined above.

Solution:
```
int makeset(int A[], int n1, int B[]) {
int i,j=0;
for(i=0;i<n1;i++) {
if(check(A[i],B,j)==0)
B[j++]=A[i];
}
return j;
}
```

(c) Complete the function `main( )`

```
int main ( ) {
        int A[100], int B[100] ;
        int i, nA, nB;
        scanf (``%d'', &nA) ;
        for (i=0; i<nA; i++)
             scanf (``%d'', &A[i]) ;
       // Call makeset
```

Solution: `nB=makeset(A,nA,B);`

```
       printarray (A, nA) ;  printarray (B, nB) ;
       return 0;
 }
```

6. (11 marks) What will be printed when the following programs/ program segments execute? Write **only** the output that will be printed if the program is executed within the box.
Marks: 3+4+4

(a)
```
#include <stdio.h>
int main()
{
    int i = 12, j, last;
    while (i > 1) {
        j = 1;
        printf("%d: ", i);

        while (j < i) {

            if ((i % j) == 0) {
                printf("%d ", j);
                last = j;
            }
            j++;
        }
        i = last;
        printf("\n");
    }
    return 0;
}
```

Solution:
```
12:  1 2 3 4 6
6:  1 2 3
3:  1
```

2 marks for guessing that, starting with 12 (counting down to 1), all factors for all numbers will be printed.
1 mark for printing all factors of 12 only

(b)
```
#include <stdio.h>
int main ()
{
  int a[] = { 6, 3, 2, 8 };
  int i, j;

  for (i = 0; i < 4; i++) {
      printf ("%d: ", a[i]);

      for (j = 0; j < 4; j++) {

          if ((a[i] % a[j]) == 0) {
              printf ("%d ", a[j]);
              continue;
            }

          if ((a[j] % a[i]) == 0) {
              printf ("%d ", a[j]);
              break;
            }
        }
        printf ("\n");
    }
  return 0;
}
```

Solution:

```
6:  6 3 2
3:  6
2:  6
8:  2 8
```

2 marks for guessing that only factors, i.e.:
6: 6 3 2
8: 2 8
2 marks guessing only the first multiple:
3: 6
2: 6

(c)
```
void serve (int num_tasks)
{
        static int server = 1;
        int taskid = 1;

        printf("Starting %d tasks\n",num_tasks);

        for (int i = 0; i < num_tasks; i++) {
                 printf (Task %d - Server %d \n", taskid, server);
                server++;
                if (server > 5)
                        server = 1;
                taskid++;
        }
        printf("Done\n");
}

int main ()
{
    serve (3);
    serve (4);
    return 0;
}
```

Solution:

```
Starting 3 tasks
Task 1 - Server 1
Task 2 - Server 2
Task 3 - Server 3
Done
Starting 4 tasks
Task 1 - Server 4
Task 2 - Server 5
Task 3 - Server 1
Task 4 - Server 2
Done
```

2 marks for getting the structure:
Starting ...
task ... - server ...
Done

1 extra mark for correct task numbers
1 extra mark for correct server numbers

7. (7 marks) Consider the following functions: **Marks:** $3+4$

(a)

```
int foo (int x, int y) {
  if (x < y)
    return x;
  else
    return foo (x - y, y);
}
```

For each call below, indicate what value is returned:

`foo (6, 13)` _______6_______

`foo (37, 10)` _______7_______

1 mark for 1st, 2 marks for 2nd

(b)

```
void baz (int n) {
  if (n <= 1)
    printf ("\%d ", n);
  else {
     baz (n/2);
     printf (",  \%d \\n",  n);
  }
}
```

For each call below, indicate what output is printed:

`baz(4)`

Solution:

```
1 , 2 \n, 4 \n
 or
1, 2
4
```

No part marks

`baz(30)`

Solution:

```
1 , 3 \n, 7 \n, 15 \n, 30 \n
 or
1 , 3
, 7
, 15
, 30
```

No part marks

[Extra Page/ Rough Work]