INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Stamp / Signature of the Invigilator

EXAMINATION (Mid Semester)	**SEMESTER (Spring)**

Roll Number										**Section**		**Name**	

Subject Number	C	S	1	0	0	0	1	**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 washroom 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**

Instructions related to this questions paper:

1. This is a QUESTION-CUM-ANSWER script.
2. This question paper has TEN questions.
3. Answer all questions.
4. All answers should be written in the blank spaces provided immediately after the question / filling in the blanks as specified. Q9 is of multiple-choice type, and the correct choice should be circled.
5. No clarifications! Make appropriate (but not unjustifiable) assumptions wherever (you feel) necessary.
6. Extra space is provided for rough work in the question paper itself. **No extra pages should be attached with this question-cum-answer script.**

1. Answer the following questions. **Marks:** $2.5+2.5+2=7$

   (a) Interpret **11011011** as a two's complement binary number, and give its decimal equivalent. Show the steps of computation.

   (b) Show how a computer would perform **10 + -3** using eight bit two's complement representation. Is there a carry? Is there an overflow? Show the steps of computation.

   (c) Find the 8-bit binary number represented by the hex number **0x3F**. It is given that this 8-bit binary number obtained is the 2's complement representation of a decimal number. Find that decimal number. Show the steps of your computation.

2. Answer the following questions. **Marks:** $4+4=8$

   (a) Convert **-35.75** to IEEE 754 floating point format. What is the hex representation of the resultant 32-bit binary number that you obtain? Show the steps of computation.

(b) A bit pattern of 32 bits when converted to hex representation gives **0x40200000**. Suppose this bit pattern corresponded to a real number represented in the IEEE-754 (single precision) floating point format. Find the decimal representation of the float number this bit pattern is representing. Show the steps of computation.

3. What will be the output of the following C Programs? **Marks:** $3+3+3=9$

(a)
```
#include <stdio.h>
int main()
{
  int a = 10, b = 20;
  int *p = &a, *q = &b;
  printf("%d ", (*p)++);
  printf("%d ", --(*q));
  printf("%d\n", a+b);
  return 0;
}
```

(b)
```
#include<stdio.h>
int *fun(int p, int *q)
{
  p = 4;
  *q = 2;
```

```
  return (q);
}
int main()
{
  int x = 6, y = 9, z = 3;
  z = *fun(x, &y);
  printf("%d %d %d\n", x, y, z);
  return 0;
}
```

(c)
```
#include<stdio.h>
int main()
{
  float arr[5] = {1.5, 2.5, 3.5, 4.5, 5.5};
  float *ptr1 = &arr[1];
  float *ptr2 = ptr1 + 3;
  float *ptr3 = ptr2 + 1;
  printf("%f ", *ptr2);
  printf("%ld ", ptr2 - ptr1);
  printf("%f\n", arr[ptr3 - ptr1]);
  return 0;
}
```

4. A lazy programmer wants to write a program that will perform two activities together – (a) reverse the elements of an integer array, and (b) sum the elements of the array. (S)he thought of writing an additional function to exchange the values of two variables. However, (s)he feels that it can be done without introducing any temporary variable within that function, exchange(int *p, int *q), which takes two pointers p and q containing the addresses/locations of two integer elements of an array and then it exchanges the values in those two locations. With this thought, (s)he started writing the program, but did not complete it. Help the lazy programmer to complete the program correctly. Each blank can have at most one statement. The part of the code is provided below. **Marks:** $1.5 \times 4 = 6$

```
#include<stdio.h>
void exchange(int *p, int *q)
{
  *p = *p + *q;

  *q = *p - *q ;

  *p = ____________________ ;
}

int main()
{
  // defining an array
  int a[] = {7,4,8,2,9,0,1,3,6,5};
```

```
  // marking the initial index of the array
  int i=0;

  // marking the final index of the array
  int f = sizeof(a)/sizeof(int) - 1;

  // initializing the sum
  int sum = 0;

  while (_____)//converge from both ends of array index
  {
    //updating sum (adding two elements of the array at a time)

    sum = _____ ;

    exchange(_____);//call to exchange function

    i = i + 1 ; // incrementing initial index

    f = f - 1 ; // decrementing final index
  }

  return 0;
}
```

5. The following program finds an approximate value of a definite integral. Let l be the left and r the right boundary for the integral. Also let h be the step size. The idea is to break the interval $[l,r]$ into sub-intervals $[l,l+h],[l+h,l+2h],[l+2h,l+3h],\ldots,[r-h,r]$. Assume that $r-l$ is an integral multiple of h. The program evaluates the function to be integrated at the center of each interval, multiplies these values by the width h and computes the sum of these products as the approximate integral. Suppose you are integrating the function x^2. Complete the following program. Each blank can have at most one statement. **Marks:** $1+2=3$

```
#include <stdio.h>
int main ()
{
    double l, r, h, x, s;
    printf("Enter left boundary : "); scanf("%lf", &l);
    printf("Enter right boundary : "); scanf("%lf", &r);
    printf("Enter step size : "); scanf("%lf", &h);
    s = 0;
    for (x = l; _____; x += h){

        s +=_____;
    }
    printf("Integral = %lf\n", s);
    return 0;
}
```

6. The following program attempts to find the roots of the polynomial $x^3-4x^2-4x+16$. It uses the fact that the product of the roots is the negative of the constant term of the polynomial. For instance, $x^3-4x^2-4x+16$ has roots, $2,-2,4$, and the product of the roots: $2\times-2\times4=-16$ (negative of the

constant term 16). To find the roots, the program checks for each integer r if it is a factor of the constant term 16, and if so, whether replacing x by the integer r in the polynomial evaluates to zero. For instance, since 4 is a root, $4^3 - 4 \times 4^2 - 4 \times 4 + 16 = 0$.

Complete the following program. Each blank can have at most one statement. **Marks:** $1+2+1=4$

```
#include <stdio.h>
int main ()
{
    int r, nroot = 0;
    for (r=1; nroot<3; r++) {
        if (16 % r == 0) {
             if (r*r*r-4r*r-4*r+16==0) {
                printf("Root found : %d\n",r);

                _____________________________;
            }
        if (_____________________________________) {
            printf("Root found : %d\n",-r);

            _____________________________;
            }
        }
    }
    return 0;
}
```

7. The following program randomly generates a sequence of integers between **-8** and **+91** and outputs the maximum and minimum values generated so far. It exits if a negative integer is generated. The program does not store the sequence generated (say, in an array), but updates the maximum and minimum values on the fly, as soon as a new entry is generated. Complete the following program. Each blank can have at most one statement. **Marks:** $1 \times 4 = 4$

   Hint: To generate a random number between 0 to n, you can use

   ```
   rand()%(n+1)
   ```

   So, for example, you can get a random number between 5 to 90 by $rand()\%86+5$

   ```
   #include <stdio.h>
   #include <stdlib.h>
   #include <time.h>
   int main ()
   {
       int a, i, max, min;
       i = 1;
       while (1) {

           a =  _____________________;
           //a stores a random number between -8 to +91
           printf("Iteration %d: new entry = %d, ", i, a);

           if (_____________________________) {
               printf("...exiting...\n\n\n");
               break;
           }
   ```

```
        if (i == 1)_____________________________;
        else {
            if (a > max) max = a;

            if (a < min) _____________________________;
        }
        printf("max = %d, min = %d\n", max, min);
        ++i;
    }
    return 0;
}
```

8. What will be the outputs of the following programs? **Marks:** $2+2=4$

(a)
```
#include<stdio.h>
int main()
{
    int x = 0, y = 10, z = 20;
    while (1) {
        x++;
        if (y > z) break;
        y += 4*x; z += 2*x;
    }
    printf("x = %d, y = %d, z = %d", x, y, z);
    return 0;
}
```

(b)
```
#include<stdio.h>
int main()
{
    int s = 0;
    while (s++ < 10) {
        if ((s < 4) && (s < 9)) continue;
        printf("%d ", s);
    }
    return 0;
}
```

9. What will be the output of the following C Programs? Only one of the 4 choices are correct. Circle only the correct choice.

Marks: $5 \times 2 = 10$

(a)
```
int fun(int *a,int n)
{
    n++;
    *a = *a + n;
```

```
    return *a;
}

int main()
{
    int n=10;
    n = fun(&n,++n);
    printf("%d",n);
    return 0;
}
```

A. 20	B. 21	C. 22	D. 23

(b)
```
#include <stdio.h>
int n = 17, r=35;
void my_disp_func( int r ) { printf("%d,%d\n",n,r); }
int main ()
{
    int n = 21, r=14;
    my_disp_func(n);
}
```

A. 17,21	B. 21,14	C. 17,35	D. 21,35

(c)
```
#include <stdio.h>
void h( int A[], int n )
{
    int i;
    for (i = 1; i < n; ++i)
        A[i] *= A[i-1];
}

int main ()
{
    int A[5] = {1,2,3,4,48};
    h(A,4);
    printf("%d", A[4]/A[3]);
    return 0;
}
```

A. 16	B. 48	C. 2	D. 4

(d)
```
#include <stdio.h>
int f(int i)
{
    return i%2;
}

int main()
{
    int i=27;
    while(f(i))
    {
        printf("%d", i);
```

```
        i = i/2;
    }
    return 0;
}
```

A. 27	B. 27 13	C. 2713	D. 1

(e)
```
#include <stdio.h>

void jumble(char A[], int size)
{
    int i;
    for(i=0;i<size;i++)
        A[i] = A[size - i -1];
}

int main()
{
    char A[] = "REMARKABLE";
    jumble(A, 10);
    printf("%s", A);
    return 0;
}
```

A. REMARKABLE	B. ELBAKRAMER
C. ELBAKKABLE	D. None of the above.

10. A natural number N is called a 'perfect number' if the divisors of N that are less than N add to N. **Example:** 6 is a perfect number because the divisors of 6 that are less than 6 (namely, 1, 2, and 3) add to 6. Given below is a program that prints all perfect numbers in a given range using functions. Fill in the blanks. Each blank can have at most one statement. **Marks:** $1 \times 5 = 5$

```
/**
 * C program to print all perfect numbers in given range using function
 */

#include <stdio.h>

/* Function declarations */
int isPerfect(int num);
void printPerfect(int start, int end);

int main()
{
    int start, end;

    /* Input lower and upper limit to print perfect numbers */
    printf("Enter lower limit to print perfect numbers: ");
    scanf("%d", &start);
    printf("Enter upper limit to print perfect numbers: ");
    scanf("%d", &end);
```

```
    printf("All perfect numbers between %d to %d are: \n", start, end);
    printPerfect(start, end);

    return 0;
}

/**
 * Check whether the given number is perfect or not.
 * Returns 1 if the number is perfect and 0 otherwise.
 */
int isPerfect(int num)
{
    int i, sum;

    /* Finds sum of all proper divisors */
    sum = 0;
    for(i=1; i<num; i++)
    {
        if(_____________________________________)
        {
            _____________________________________;
        }
    }

    if(_____________________________________)
        return 1;
    else
        return 0;
}

/**
 * Print all perfect numbers between given range start and end.
 */
void printPerfect(int start, int end)
{
    /* Iterates from start to end */
    while(start <= end)
    {
        if(_____________________________________)
        {
            printf("%d, ", start);
        }

        _____________________________________;
    }
}
```

[Extra Page/ Rough Work]

[Extra Page/ Rough Work]

[Extra Page/ Rough Work]