	INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR	**Stamp/Signature of the Invigilator**

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

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 writing examination.
4. Data book, codes, graph papers, relevant standard tables/charts or any other materials are allowed only under the instruction from the paper-setter.
5. Use the 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. **You must clearly write your roll number, name and section (11 to 20) on this front page of the answer book.**
7. **Write the answers only on the space provided and do not tear off any page. Use the last page(s) of the answer script for rough work.** Report to the invigilator if the answer script has torn or distorted page(s).
8. 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.
9. 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 consumption of any kind of beverages are strictly prohibited inside the examination hall.
10. 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.
11. 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 Scrutinizer**

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Date: 25/11/2019 AN **Time: 3 hours** **Full marks: 100** **No. of students: 800 (approx.)**

Autumn End Semester Exams, 2019 **Dept: Computer Science and Engineering** **Sub No: CS10001**

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

Instructions: ***Answer all questions in the spaces (box or fill-in-the-blank) provided. For rough work, you may use the extra pages provided in this booklet. No other supplementary sheets will be provided.***

Q1. Answer the following. **[Total: 25 marks]**

(a) What would be printed after the execution of the following code segment? **(2 marks)**

```
int i=0, x=1;
if (i++) printf ("%d", x-i);
else     printf ("%d", x+i);
```

Q1(a):

(b) What would be printed after the execution of the following code segment? **(2 marks)**

```
int i=4;
switch (i) {
  case 0:  printf ("H2");
  case 1:  printf ("O2"); break;
  case 2:  printf ("N2");
  case 3:  printf ("Fe"); break;
  default: printf ("CO2");
}
```

Q1(b):

(c) What would be printed after the execution of the following code segment? Note that there is always a blank space between two words. **(2 marks)**

```
char line[] = "A cunning fox is running.";
char *p;
p = line + 3;
printf ("%s", p+7);
```

Q1(c):

(d) Consider that a stack of positive integers is implemented with **PUSH(int x)** and **POP()** operations. If the stack is empty, the **POP()** operation will return **-1**. Assume that the stack is initially empty and following sequence of operations is performed: **(2 marks)**

```
{
  PUSH(20); PUSH(40); PUSH(30);
  POP(); POP(); POP(); POP();
  PUSH(50); PUSH(60); PUSH(10);
  POP(); POP();
}
```

Q1(d):

What will be the value returned from the stack if the next operation is **POP()** ?

(e) Consider the following recursive function: **(2 marks)**

```
int f (int n)  {
   if (n==0) return 1;
   else if( n==1) return 1;
   else return (n * f(n-2));
}
```

Which one of the following would be the result of `f(n)*f(n-1)` ?

(i) `n! (n-1)!`
(ii) `n!`
(iii) `(n-1)!`
(iv) `None of the above`

Q1(e):

(f) If the following is a valid C code, what will be printed by it? If it is invalid, what is the reason?

```
int c[20] = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19};
int (*guess)[4] = (int (*)[4])(c+5);
printf ("guess[2][3]: %d\n", guess[2][3]);
```

(2 marks)

Q1(f):

(g) Consider the following list stored in an array maintaining the same order of the values from its lowest index to the highest index of storage.

`10, 20, 30, 40, 50, 55, 60, 70, 80, 90, 100`

Given a value 82, find the values in the array to be compared with 82 if binary search is applied?

(2 marks)

Q1(g):

(h) Consider the following list stored in an array in the given order. Its first element is used as the pivot for partitioning the array for sorting the list in ascending order using quick sort. Show the array after ***first partitioning*** with respect to the pivot element 33. **(2 marks)**

`33, 25, 67, 89, 90, 24, 16, 35`

Q1(h):

(i) What is printed by the program below? **(4 marks)**

```
#include <stdio.h>
int scan (int *a, int m)
{
  static int rev=0;
  int j,k;
  for (j=0;j<m-1;j++)
    if (a[j]>a[j+1]) {rev++; k=a[j]; a[j]=a[j+1]; a[j+1]=k;};
  return(rev);
}
int main ()
{
  int i, n=6;
  int a[20] = {6, 5, 4, 3, 2, 1};
  for (i=0;i<n-1;i++)
    printf ("%d ", scan(a,n-i));
  printf("\n");
}
```

Q1(i):

(j) What does the following program print? **(3 marks)**

```
#include <stdio.h>
int main() {
  int i, p[][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};
  for (i=0;i<3;i++)
     printf("\n (%d, %d) \n", *p[i], (*p)[i]);
}
```

Q1(k):

(k) Study the following function and count the number of times it is called and the value returned when it is invoked as **`guess(5)`**. **(2 marks)**

```
int guess (int m)
{
  int temp;
  if (m < 2) return 1;
  else temp = 5*guess(m-1) + 3*guess(m-2);
  return temp;
}
```

Q1(l):

Times called:

Value returned:

Q2. Answer the following. **[Total: 4+3+4+4=15 marks]**

(a) State the following ***in decimal***: **(4 marks)**

	Minimum	Maximum
(i) Minimum and maximum signed numbers that can be represented in 12-bit 2's complement		
(ii) Minimum and maximum signed numbers that can be represented in 10-bit 1's complement		

(b) Convert the decimal number $(239.39)_{10}$ to binary, with seven bits in the fractional part. **Only write the final result in the box below**, and show the calculations in the rough page. **(3 marks)**

2(b):

(c) The following code segment converts a non-zero positive decimal integer into a character string containing the hexadecimal equivalent. The program repeatedly divides the given number by 16, and accumulates the remainders in reverse order. Fill in the blanks with correct code segments. **(4 marks)**

```
#include <stdio.h>
#include <string.h>
void Dec2Hex (int n, char hexstr[])  {
      int k, hex = 0;
      if (n == 0)  return;
      else {
         hex = _____________________;

         Dec2Hex (_____________________________);
         k = strlen(hexstr);
         if (hex > 9) {
            hexstr[k] = _____________________________;
            hexstr[k+1] = '\0';
         }
         else {
            hexstr[k] = _____________________________;
            hexstr[k+1] = '\0';
         }
}
int main()  {
      int dec_number;  char hex_number[10] = "";
      printf ("\nEnter a nonzero decimal number: ");
      scanf ("%d", &dec_number);
      Dec2Hex (dec_number, hex_number);
      printf ("\nHexadecimal equivalent: %s", hex_number);
}
```

(d) Represent **+78** and **-116** in 8-bit 1's complement representation, and carry out the arithmetic operation **(78 – 116)** using 1's complement arithmetic, showing the steps of computation. You have to carry out the subtraction using addition operation only. **(4 marks)**

2(d):

Final answer (in 1's complement):

Q3. Answer the following. **[Total: 4+4+4+3=15 marks]**

(a) The following binary search function takes an 1-D array **Data[]**, the number of elements **n**, and an element to search **mynum** as parameters, and returns the array index if the number is found; and returns -1 otherwise. Fill in the blanks with correct code segments. **(4 marks)**

```
int binary_search (float Data[], int n, float mynum)
{
   int  left, right, middle, flag;
   left = 0;  right = n - 1;

   while  (_____________________________)  {
     middle = (left + right) / 2;
     flag = (Data[middle] < mynum);
     if  (_____________________________________) return middle;

     else if  (flag)  _____________________________________;

     else             _____________________________________;
   }
  return  -1;    /* NOT FOUND */
}
```

(b) Consider the following program for sorting an array of student records in ascending order by their roll numbers using bubble sort. Fill in the blanks with correct code segments. **(4 marks)**

```
#include <stdio.h>
typedef struct student_record {
   int     rollno;
   char    name[30];
   float   cgpa;
} stud;

int main()
{
   stud rec[50], temp;
   int i, j, n;
   printf ("\nEnter number of students: ");
     scanf ("%d", &n);
   printf ("\nEnter the student records one by one:\n");
   for (i=0; i<n; i++)
      scanf ("%d %f %[^\n]", &rec[i].rollno, &rec[i].cgpa,
                                        rec[i].name);
     for (_________________; i>0; i--)

       for (j=0; _____________; j++)

         if (_____________________________________________)  {
              temp = rec[j];
              rec[j] = rec[j+1];

              _____________________________;
          }
     printf ("\nStudents in order of roll number:");
     for (i=0; i<n; i++)
       printf ("\nRoll: %d, Name: %s, CGPA: %f",
                  rec[i].rollno, rec[i].name, rec[i].cgpa);
}
```

(c) Fill up the following table with correct values. **(4 marks)**

Maximum number of comparisons required to search for an element in a sorted array with **N** elements using binary search:	
Maximum number of comparisons required to merge two sorted arrays with **M** and **N** elements respectively:	
Minimum and Maximum number of comparisons required to search for an element in a unsorted array with **N** elements using linear search:	
Average number of comparisons required to sort a list of numbers with **N** elements using selection sort:	

(d) For the given list of 7 numbers, show how the numbers get rearranged after every iteration of the insertion sort algorithm: **(3 marks)**

Input list	40 10 30 50 70 25 60 5
After iteration 1	
After iteration 2	
After iteration 3	
After iteration 4	
After iteration 5	
After iteration 6	
After iteration 7	

Q4. Answer the following. **[Total: 5+5+5 = 15 marks]**

(a) Consider the following code snippet: **(2.5+2.5 = 5 marks)**

```
int rec(char str[], int s, int e)
{
   if (s == e) return 1;
   if (str[s] != str[e]) return 0;
   if (s < e + 1)  return rec(str, s + 1, e - 1);
   return 1;
}

int check(char str[])
{
   int n = strlen(str);
   if (n == 0) return 0;
   if (rec(str, 0, n - 1) == 1)  return 1;
   else  return 0;
}
```

i) What will `check("371")` return?

ii) What will `check("111")` return?

Q4(a)-(i):

Q4(a)-(ii):

(b) The following program should take an input word and then reverse it. For example, if the input word is "**iitkgp**", then the reverse will be "**pgktii**". Find the errors (both syntactic and logical) in the program, and mention the line numbers and corresponding correct versions. **(5 marks)**

```
1: #include <stdio.h>
2: #include <string.h>

3: void reverse(char, int, int);

4: int main()
5: {
6:     char a[100];
7:     scanf ("%s", a);
8:     reverse (a, 0, strlen(a)-1);
9:     printf ("%s\n", a);
10:    return 0;
11:}


12: void reverse(char x, int begin,
                  int end)
13: {
14:     char c;

15:     if (begin <= end)  return;
16:     c  = *(x+begin);
17:     *(x+begin) = *(x+end);
18:     *(x+end) = c;

19:     reverse(x, begin, end);
20: }
```

Line No.	Correct version

(c) The following program finds out the initials of the words in a string, where a single blank space separates two consecutive words in the string. For example, is the input string is "**swami vivekananda**", the output will be "**sv**". Fill up the blanks in the program. **(5 marks)**

```
#include <stdio.h>

void initials (__________);        // Prototype declaration

int main() {
   char str[20];
   int i=0;
   printf ("Enter a string: ");

   scanf ("%[________\n]", str);   // Read a line
   printf ("%c", *str);            // Print first character

   initials (__________);          // Function call
   return 0;
}
```

```
void initials (char* str)
{
    int i = 0;
    while (str[i] != _____ ) {    // Termination condition
       if (str[i]==' ') {
          i++;
          printf ("%c",*(_____));  // Output
       }
       i++;
    }
}
```

Q5. Answer the following. **[Total: 8+3+4 = 15 marks]**

(a) The following C program accepts three command-line arguments in the following order – (i) a source file, (ii) a search-string, and (iii) a destination file. The program opens the files, and reads the source file line by line. For each line, it checks if the search-string is contained within the line (using a function called '**match**'). If yes, then this line is written into destination file. Finally, both files are closed. It is assumed that a line in the file is at most 200 characters long. Fill in the blanks in the code, so that the program works correctly. **(8 x 1 = 8 marks)**

```
#include <stdio.h>
#include <string.h>
#define MAXLEN 200

// function to check if srchstr appears in line
// returns 1 if match found (srchstr appears in line), 0 otherwise
int match(char *line, char *srchstr)
{
    int lenline, lensrch, i, j;

    lenline = strlen(line);
    line[--lenline] = '\0';    // Remove trailing newline from line
    lensrch = strlen(srchstr);
    for (i = 0; i <= lenline - lensrch; i++) {
       for (j = 0; j < lensrch; j++) {

            if (_____)  break;
                          // Character in two strings did not match
       }
       if (_____) {
                          // srchstr found at position i of line
            return 1;
       }
    }
    return 0;    // No match found
}

int main(int argc, char *argv[])
{
    char srchstr[MAXLEN], line[MAXLEN];
    FILE *fin, *fout;
```

```
    // check if correct number of command-line args are given
    if (_________________) {
       printf("Usage: ./a.out <infile> <search string> <outfile>");
       exit(1);
    }

    fin = fopen(_____________________);     // Open source file

    fout = fopen(_____________________);    // Open destination file

    // check if files opened correctly

    if (_____________________________) {
       printf ("Could not open at least one of the files");
       exit(1);
    }

    strcpy (srchstr, argv[2]);

    while (fgets(_____________________  ) != NULL) {
                        // Read source file line by line, until EOF

       if (match(line, srchstr) == 1) {
                        // Check if srchstr appears within line

             fprintf (_____________________________);
                         // Write line to destination file
       }
    }
    fclose(fin);
    fclose(fout);
    return 0;
}
```

(b) Consider the following three functions: **(3 marks)**

```
int* F1(void) {
  int x = 10;
  return (&x);
}

int* F2(void) {
  int *px;
  px = (int *) malloc (sizeof(int));
  *px= 10;
  return px;
}

int* F3(void) {
  int *px;
  *px = 10;
  return px;
}
```

Which of the above three functions is/are likely to cause problems with pointers, when executed? Select the correct option.

A. Only F1
B. Only F2
C. Only F3
D. F1 and F2, but not F3
E. F1 and F3, but not F2
F. F2 and F3, but not F1

Q5(b):

(c) Write whether the following statements are **True** or **False**. **(4 x 1 = 4 marks)**

(i) Elements of a dynamically allocated 2-D array may not be contiguous in memory.

(ii) Elements of a dynamically allocated 1-D array may not be contiguous in memory.

(iii) Global variables and dynamically allocated variables are stored in heap segment of memory.

(iv) The statements **`calloc(m,n)`** and **`malloc(m*n)`** allocate the same number of bytes.

(i)	
(ii)	
(iii)	
(iv)	

Q6. **Answer the following.** **[Total: 2+3+4+4+2 = 15 marks]**

(a) What will be the output of the following program? **(2 marks)**

```
#include <stdio.h>
struct student {
        char name[20];
        int age;
    };
struct student  std1={"Amit", 20}, std2;

int main()
  {
    std2 = std1;
    std2.name[1] = 'j';
    printf("%s, %d", std2.name, std2.age);
    return 0;
  }
```

Q6(a):

(b) For a given linked list: "**`10 --> 20 --> 30 --> 40 --> 50`**", what will be the output when **`funcLL(head)`** is called, where **`head`** is a pointer that points to the first node? **(3 marks)**

```
void funcLL (struct node* head)
{
  if (head == NULL)
    return;

  funcLL (head->next);
  printf ("%d  ", head->data);
}
```

Q6(b):

Assume the following structure definition:

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

(c) Consider the function n**odeCount** to count the number of nodes in a Linked List (with first node pointed to by "**head**"). Fill in the blanks in the code, so that the function works correctly. **(4 marks)**

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

int nodeCount(struct node *head)
{
    int count = _______________;  // Initialize count
    struct node *ptr;

    ptr = ___________________;

   while (ptr != _____________)
    {
        count++;

        ___________________________ ;
    }
    return count;
}
```

(d) For a given linked list: "**1 --> 2 --> 3 --> 4 --> 5 --> 6**" , what will be the output when **newfuncLL(head)** is called, where **head** is a pointer that points to the first node? **(4 marks)**

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

Q6(d):

```
void newfuncLL (struct node *head)
{
  if (head == NULL) return;
  printf ("%d  ", head->data);

  if (start->next != NULL )  newfuncLL (head->next->next);
  printf("%d  ", head->data);
}
```

(e) Consider the following program to find the Euclidian distance between two points in a 2D plane. Identify the errors in the code and write the corresponding correct statements. **(2 marks)**

Q6(e):

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

struct Point {
    float x, y;
    }  P1, *P2;

int main()
{
    float dist;
    P2 = (struct Point *) malloc (sizeof(struct Point));
    printf ("Point-1: Enter x and y: ");
    scanf ("%f %f", &P1.x, &P1->y);

    printf("Point-2: Enter x and y: ");
    scanf("%f %f", &P2.x, &P2->y);

   // Calculate distance between two points
   dist = sqrt ((P1.x - P2->x)*(P1.x - P2->x)
                          + (P1.y - P2->y)*(P1.y - P2->y))
   printf ("\nDistance = %f\n", dist);
   return 0;
}
```

ROUGH WORK

ROUGH WORK

ROUGH WORK