Indian Institute of Technology Kharagpur

Stamp/Signature of the Invigilator

EXAMINATION ANSWERSCRIPT

Autumn Semester 2025-26 – Mid Sem Exam	**SEMESTER (Autumn)**

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

Subject Number	C	S	1	0	0	0	3		**Subject Name**	Programming and Data Structures

Department/Centre/School		**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, the 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 your 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**'. Don't adopt unfair means and also don't indulge in unseemly behavior.

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

Signature of the Student

To be Filled 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**

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Mid-Autumn Semester Examination 2025-26

Duration: 2 hrs. **Full Marks: 50**

Subject No.: CS10003 **Subject: Programming and Data Structures**

Department/Center/School: Computer Science and Engineering

Specific charts, graph paper, log book etc., required NONE

Special Instructions (if any): No clarification will be provided during the exam. State any assumption made.

Your final answer written in the space provided with each question will only be evaluated. Also, do not show any rough work inside those spaces.

Do not write anything on this page

Q.1. Answer the following questions as directed. **[10×2 = 20 Marks]**

a. What will get displayed when the following program is executed?

```
#include <stdio.h>
int main() {
      int k;
      for (k = 1; k = -1; k++)
            if (k < 6) break;
      printf("%d\n", k);
      return 0;
}
```

-1

b. What will get displayed when the following program is executed?

```
#include <stdio.h>
int main() {
      int x = 6, y = 12, result = 15;
      result += x/5 *13 *y/3 *x;
      printf("result=%d\n", result);
      return 0;
}
```

result=327

c. What will get displayed when the following program is executed?

```
#include <stdio.h>
int main() {
      char b = 'b';
      while ((b >= 'b') && (b <= 'd')) b++;
      printf("%c\n", b);
      return 0;
}
```

e

d. What value will the following function return when called as `recur(4)`?

```
int recur(int data) {
if (data > 2)
return (recur(data - 1) - recur(data - 2));
else return 1;
}
```

-1

e. What will get displayed when the following program is executed?

```
#include <stdio.h>
int main() {
      int x = -8, y = 7; if (x > y) x = 1;
      else if (y < 0) x = x*(-1);
      else x = 2*x;
      printf("x=%d\n", x);
      return 0;
}
```

x=-16

f. What will get displayed when the following program is executed?

```
#include <stdio.h>

void f1(int b[], int k){
        int i;
        for(i=0;i<k;i++) b[k]=b[k-1]+b[k-2];
}

void f2(int c[], int k){
        int i;
        for(i=2; i<k; i++) f1(c, i);
}

int main() {
        int a[5] = {1, 2, 3, 4, 5};
        f2(a, 5); printf("%d", a[4]);
        return 0;
}
```

8

g. What will get displayed when the following program is executed?

```
#include <stdio.h>
int var = 5, var2 = 10;

int f(int var3) {
        int var = 10;
        var += var2;  var2 += var3;
        return var;
}

int main() {
        int var2 = 15; printf("%d, %d", f(var2), var2);
        return 0;
}
```

20, 15

h. What will get displayed when the following program is executed?

```
#include <stdio.h>

void increment(int i) { i++; }

int main() {
      int i = 0, j = 0;
      while (i++ < 10) increment(j);
      printf("i=%d, j=%d\n", i, j);
      return 0;
}
```

i=11, j=0

i. What will get displayed when the following program is executed?

```
#include <stdio.h>
int main() {
      float j=1.0, i=2.0; int n=0;
      while (i/j > 0.05) {
            j += j; n++;
      }
      printf("n=%d, j=%f\n", n, j);
      return 0;
}
```

n=6, j=64.000000

j. What will get displayed when the following function is called as `f(2, 10)`?

```
int f(int x, int y) {
      int sum = 0; y--;
      if (x == 0) return 0;
      else {
               printf("%d:", x);
               sum = y + f(x – 1, y);
               printf ("%d:", sum);
            }
      return sum;
}
```

2:1:8:17:

Q.2. Answer the following questions as directed.

a. Consider the following program. Fill in the blank to complete the macro such that it satisfies the sample inputs and outputs. **[2 Marks]**

```
#include <stdio.h>

#define MIN(A, B) (A < B ? A : B)
int main() {
    int x, y;
    scanf("%d %d",&x, &y); printf("%d\n", 1 + MIN(x, y));
    return 0;
}
```

Inputs	X = 10, y = 20	X = 20, y = 10	X = 20, y = 20
Outputs	11	11	21

b. What will get displayed when the following program is executed? **[3 Marks]**

```
#include <stdio.h>
void alpha(int n); void beta(int n);

void alpha(int n) {
    if (n > 0) { printf("%d ", n);
     beta(n - 1); } }

void beta(int n) {
    if (n > 0) { printf("* ");
     alpha(n / 2); } }

int main() { alpha(8);  return 0; }
```

8 * 3 * 1

c. Consider the recursive function `count` that counts the number of times a specific character `ch` Appears in a string `str`. For example, `count("IITKHARAGPUR", 0, 'A')` should return 2. Fill in the two blanks to complete the function. **[3 Marks]**

```
int count(char str[], int i, char ch) {

  if (str[i] == '\0') return 0; (Base case)
  // Recursive call
  return (str[i] == ch?1:0) + count(str, i+1, ch);
  }
```

d. Consider the program segment given below to read a letter from `a..z` and `A..Z` from the keyboard and convert it to lowercase if not already so. It is assumed that the user will only input a character from `a..z` and `A..Z`. Fill in the blank with a single C expression so that the variable `ch` will contain the input character in lowercase. Do not use any library function. **[2 Marks]**

```
char ch; ch = getchar();

ch = (ch>='A')&&(ch<='Z')? 'a'+ch-'A':ch;
```

e. What will get displayed when the following program is executed? **[2 Marks]**

```
#include<stdio.h>
void remove_duplicates(char word[]) {
     int k, j; char prev = '\0';
     for (k = j = 0; word[k] != '\0'; k++) {
     if (prev != word[k])
     word[j++] = word[k]; prev = word[k];
     }
     //word[ j ] = '\0';
}

int main() {
     char inpstring[12] = "Mississippi";
     remove_duplicates(inpstring); printf ("%s",inpstring);
     return 0;
}
```

Misisipippi

f. The following recursive C function `move` takes an integer array `A` and two integers, `leftIndex` and `rightIndex` as input parameters. The goal of the function is to rearrange the subarray between `leftIndex` and `rightIndex` so that all even numbers are moved to the left and all odd numbers are moved to the right. The order of elements within the even or odd sections does not matter.

For example, if `A = {3, 8, 5, 6, 7, 4, 9}` and the call is `move(A, 1, 5)`, a valid result would be `A = {3, 8, 4, 6, 7, 5, 9}`.

Fill in the three blanks to implement the function correctly. **Each blank will contain one single statement**. **[3 Marks]**

```
void move(int A[], int leftIndex, int rightIndex) {
  if (leftIndex < rightIndex) {
    // CASE 1: The number on the left is EVEN
    if (A[leftIndex] % 2 == 0)

       move(A, leftIndex + 1, rightIndex);

    // CASE 2: The number on the right is ODD
    else if (A[rightIndex] % 2 != 0)

      move(A, leftIndex, rightIndex - 1);

    // CASE 3: The left is ODD and the right is EVEN.
    // This 'else' only runs if both Case 1 and Case 2 are false.
    else {
      int temp = A[leftIndex]; A[leftIndex] = A[rightIndex];
      A[rightIndex] = temp;

      move(A, leftIndex + 1, rightIndex - 1);
    }
  }
}
```

g. The following function `strEqual` takes two strings *S*1 and *S*2 as parameters. Fill in the three banks in the function so that it returns `1` if the two strings are the same, `0` otherwise. **[3 Marks]**

```
int strEqual ( char s1[], char s2[] ) {

   int i = 0;
    /* Go on until the end of either of the strings */

    while ( s1[i] != '\0' && s2[i] != '\0' )

      if (S1[i] != S2[i])  return 0; i++;
   }

   if ( s1[i] == '\0' && s2[i] =='\0' )
     return 1;
   else return 0;
  }
```

h. Consider the following recursive function named `sum`.

```
int sum(int x)
{
      if(x==1) return 1;
      printf("SUM\n");
      return(sum(x--));
}
```

How many times will the word `SUM` be printed if the function is called as `sum(4)`? Write only the correct option number in the box provided. **[2 Marks]**

(i) Less than 3 times
(ii) Exactly 3 times
(iii) Exactly 4 times
(iv) More than 4 times

(iv)

3. Consider the following C program. Given a single ODD integer `N` (where $1 \leq N \leq 9$), it prints an `N×N` grid filled with the integers from 1 to N^2 arranged in a counter-clockwise spiral starting from the centre. The spiral rules are as follows:

i. Place 1 at the exact centre of the grid (row `N`/2, column `N`/2 using 0-based indexing).
ii. From there, move right, up, left, down, right, up, and so on, expanding in a counter-clockwise spiral until the grid is filled.
iii. Print the grid row by row.
iv. Each number must be right-aligned to the width of the largest number (N^2).

Example: For `N`=5 and `N` =9, the output spirals will be as follows.

```
17 16 15 14 13
18  5  4  3 12
19  6  1  2 11
20  7  8  9 10
21 22 23 24 25
```

Output for N=5

```
65 64 63 62 61 60 59 58 57
66 37 36 35 34 33 32 31 56
67 38 17 16 15 14 13 30 55
68 39 18  5  4  3 12 29 54
69 40 19  6  1  2 11 28 53
70 41 20  7  8  9 10 27 52
71 42 21 22 23 24 25 26 51
72 43 44 45 46 47 48 49 50
73 74 75 76 77 78 79 80 81
```

Output for N=9

Fill in the ten blanks so that the program generates the desired output. **[10×1=10 Marks]**

[The code is shown in the next page.]

```
#include <stdio.h>
#define IDX(r, c) (r * N + c)

int main() {

      int N, num, step, dir, maxNum, r, c, rep, i, j;
      int grid[100];
      int dr[4] = {0, -1, 0, 1};
      int dc[4] = {1, 0, -1, 0};
      scanf("%d", &N);

      if (N < 1 || N > 10 || ( N%2==0 ) )
            return 0;

      r = N / 2; c = N / 2;
      num = 1; step = 1;  dir = 0;

      grid[ IDX(r, c) ] = num++;

      while (num <= N * N) {
         for (rep = 0; rep < 2; rep++) {

             for (i = 0; i < step && num <= N * N ; i++)  {


                 r += dr[dir] ;

                 c += dc[dir] ;

                 grid[IDX(r, c)] = num++ ;
             }

             dir = (dir + 1) % 4 ;
         }
         step++;
      }

     maxNum = N * N ;

    for (i = 0; i < N; i++) {
        for (j = 0; j < N; j++) {

            printf("%2d", grid[IDX(i, j)]);

            if ( j < N - 1 ) printf(" ");
        }
        printf("\n");
    }
    return 0;
}
```

Space for Rough Work

Space for Rough Work

Space for Rough Work