INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Stamp / Signature of the Invigilator

EXAMINATION: Mid Semester (Autumn 2023-24)	**Answer all (Duration: 2 hours)**

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 the answer-script and do not tear off any page. **For rough work, use last page(s) of the answer script and white spaces around the questions.** 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**	**10–14**	**Total**
Marks Obtained											

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

The end of this question paper is marked by — END —.
Write the answers *only in the designated boxes or above the blank lines*.

1. What will be printed by the following program? **[5]**

```
#include <stdio.h>

int main ()
{
    char c1 = 'A', c2 = 'Z'; /* ASCII value of 'A' is 65 */
    char c3 = c2 - c1 + 45;
    int i = (c1 < c3) && (c3 > '\0');
    int j = (c3 != c2) || ((c3 - c2) > c1);
    int k = (i != j) + (c3 > 100) - (c3 < 80);
    printf("c3 = %c c3 = %d i = %d j = %d k = %d", c3, c3, i, j, k);

    return 0;
}
```

```
c3 = F c3 = 70 i = 1 j = 1 k = -1
```

2. Consider the following code. **[1+2+1+1=5]**

```
#include <stdio.h>

int main(){
    int i, sum = 0,count=0;

    printf("Numbers: ");
    for(i = 1; !((i+1)%5==0 || (i+7)%6 == 1 ||
                    (i-27)%7==2); i = i*(i+3)/2, count++){
        printf("%d ", i%17);
        sum += i/12345678;
    }
    printf("\n");
    printf("Count = %d\n",count);
    printf("Sum = %d\n", sum);
    return 0;
}
```

Does the loop terminate (yes/no)? _____Yes

What will be the output of the above code?

```
Numbers: 1 2 5 3 9 3 9
Count = 7
Sum = 29
```

3. Given an array of n integers and an integer k, the following code determines whether there are two elements in the array such that their difference is k. If found, it prints their indices. Fill up the following code so that it produces the correct result. Assume that $2 \leq n \leq 100$. **[5]**

```
#include <stdio.h>

int main(){
  int a[100], n, k, i, j, found = 0;

  printf("Enter n & k: "); scanf("%d%d", ___&n, &k___); // 1 mark
  printf("Enter the elements: ");
  for(i=0; i<n; i++)
    scanf("%d", &a[i]);

  for(i=1; i<n && !(found); i++)
    for(j=0; j<i && !(found); j++)

      if(____(a[i]-a[j]==k) || (a[j]-a[i]==k)_________) // 2 marks
        found = 1;
  if(found)
    printf("Found indices: %d,%d.\n", ___i-1, j-1___); // 2 marks
  return 0;
}
```

4. A and B are two points on the plane whose coordinates are integers and given as input. We have to find the number of points whose coordinates are integers and which lie on the straight line segment $\overline{\text{AB}}$. For example, if $\text{A} = (-2,3)$ and $\text{B} = (2,9)$, then the answer is 3 because the points with integer coordinates that lie on $\overline{\text{AB}}$ are $(-2,3), (0,6), (2,9)$ and none else. Fill up the following code so that it produces the correct result. **[10]**

```
#include <stdio.h>

int main(){
  int x1, y1, x2, y2, a, b, c;
  printf("Enter coordinates of A and B: ");

  scanf("____%d%d%d%d____", &x1, &y1, &x2, &y2); // 2 marks

  a = (x1 < x2) ? (x2 - x1) : ____(x1 - x2)____; // 1 mark

  b = (y1 < y2) ? (y2 - y1) : ____(y1 - y2)_____; // 1 mark

  while(___b !=0____){ // 2 marks

    ___c = b___; // 2 marks

    b = a%b;

    ___a = c___; // 2 marks
  }
  printf("Number of points = %d.\n", a+1);
  return 0;
}
```

5. The following program populates the first n ($1 \leq n \leq 100$) prime numbers in an array, using the facts that (i) 2 is the smallest prime and (ii) a number is *prime* if and only if it is not divisible by any prime less than its square root. Every newly discovered prime, obtained this way, is put in the array. Fill in the blanks. **[10]**

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

int main() {
  int n, i=2, tag, j=0, number, _PRIME[100], limit;

  _PRIME[j] = _________2_________;                    // 1 mark
  printf("How many primes? ");
  scanf ("%d", &n);

  for(number = ____PRIME[j]+1_______; ; number++){   // 1 mark

    limit = __________floor(sqrt(number))_________;  // 2 marks
    tag = 0;
    for (i=0; _PRIME[i] <= limit; i++){

      if (number % ______PRIME[i]______ == 0){       // 2 marks
        tag = 1;
        break;
      }
    }

    if (!(_______tag________)){                      // 1 mark

        _________j++_______; _PRIME[j] = number;}    // 1 mark

    if (j > n-1) break; // 1 mark
  }
  printf("The first %d primes: ", n);
  for(i=0; i<j; i++)
    printf("%d ", _PRIME[i]);
  return 0;
}
```

Write the output when $n = 8$. // 2 marks

```
The first 8 primes: 2 3 5 7 11 13 17 19
```

6. Fill in the one blank line given and write the output (five lines) of the program when input = 2. **[5]**

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

_______float myfunc(float);_______ // Function declaration: 2 marks

int main(){
  float r, area, a = 10.0;
  scanf ("%f", &r);
  printf ("A = %f\n", a);
  area = cArea(cArea(r));
  printf ("A = %f\n", a);
  printf ("Area is %f \n", area);
  return 0;
}

/* Function to compute the area of a circle */
float cArea(float r){
  printf("r = %f\n", r);
  return 3.14159 * r * r;
}
```

```
A = 10.000000
r = 2.000000
r = 12.566360
A = 10.000000
Area is 496.099213
```

7. The school-book multiplication algorithm for two non-negative decimal integers a and b, each possibly having multiple digits, can be written as:

$$a \times b = a \cdot b_0 \cdot 10^0 + a \cdot b_1 \cdot 10^1 + \cdots + a \cdot b_{n-1} \cdot 10^{n-1},$$

where the number b is represented in terms of its n decimal digits as $b_{n-1} \cdots b_1 b_0$.
For example, if $a = 413$ and $b = 7685$, then

$$a \times b = 413 \cdot 5 \cdot 1 + 413 \cdot 8 \cdot 10 + 413 \cdot 6 \cdot 100 + 413 \cdot 7 \cdot 1000.$$

Fill in the blanks of the following program to compute and print the product of two non-negative integers following the above algorithm. The two non-negative integers are input by the user. **[6]**

```
// Each blank carries 1 mark
#include <stdio.h>
int main(){
  // declare non-negative integer variables

  unsigned a, b, weight=1, product;

  printf("\nEnter the values of a and b (non-negative): ");
  scanf("%u%u", &a, &b);

  product = 0; // initial value

  // Perform the iterative multiplication
  while (b > 0) {
    product += a * (b % 10) * weight; // update product

    b = b / 10; // update b

    weight = weight * 10; // update place value
  } // end while

  printf("\nProduct = %u\n", product); // Print the product
  return 0;
}
```

8. Fill in the blanks to calculate the value of the power-series $e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$. Once the user enters the value of x, the calculations continue till the difference (returned by `fabs()`) between the calculated value and the value returned by `exp()` is not less than 10^{-4}. The functions `fabs()` and `exp()` are used from the math library. **[4]**

```
// Each blank carries 1 mark
#include <stdio.h>
#include <math.h> // for exp() and fabs()

int main(){
    double x, term, val;
    int n;
    printf("\nEnter the value of x: ");
    scanf("%lf", &x);

    val = 1.0; // value of the power-series
    term = 1.0; // value of a single term of the power-series
    n = 1; // iteration number

    while (fabs(val - exp(x)) >= 1e-4) {

        term = term * x / n;

        val += term;  n = n + 1;
    }
    printf("\nexp(%lf): %lf\n", x, val);
    return 0;
}
```

9. Write the output of the following code. **[3 + 2]**

```
#include <stdio.h>
int server = 1, taskid = 1;
void serve (int num_tasks){
  int taskid = 1;
  for (int i = 0; i < num_tasks; i++) {
    printf("Task %d - Server %d \n", taskid, server);
    server++; taskid++;
    if (server > 3) server = 1;
  }
}

int main(){
     serve(2);
     serve(3);
}
```

```
Task 1 - Server 1
Task 2 - Server 2
Task 1 - Server 3
Task 2 - Server 1
Task 3 - Server 2
```

Suppose the `serve()` function in the above code prints the tasks executed on different servers. Suggest a change in **only one line of the code** so that the `serve()` function prints the total number of tasks executed till that time. Write that line of code before the change and the same after the change.

Before change: `int taskid = 1;`

After change: `// int taskid = 1;`

10. Consider the following function which calculates the *mode* of a given integer array (`a[]`), passed as an argument along with the size (`n`). [*Mode* is an element that appears the maximum number of times. Ties are broken arbitrarily.] Fill in the blanks to complete the code. **[5]**

```
// Each blank carries 1 mark

int mode(int a[],int n) {
   int maxValue, maxCount, i, j,count;

   _____; // initialization of maxCount
```

```
    for (i = 0; i < n; ++i) {

        ____________________;

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

            if (____________________)
            ++count;
        }
        if (count > maxCount) {

            ____________________;

            ____________________;
        }
    }
    return maxValue;
}
```

Blank 1: maxCount=0;
Blank 2: count = 0;
Blank 3: a[j] == a[i]
Blank 4: maxCount = count;
Blank 5: maxValue = a[i];

11. Determine the single-precision floating-point representation from the given normalized number $-1.111100011001 \times 2^{56}$. **[2]**

```
Sign bit = 1 (for -ve numbers) (1 bit)
Exponent = E - 127 = 56 -> E = 183 -> 10110111 (8 bits)
Mantissa = 11110001100100000000000 (23 bits)
Single-precision floating-point representation = 32 bits
sign bit|Exponent|Mantissa = 11011011111110001100100000000000
```

12. Print the output of the following program. **[3]**

```
#include <stdio.h>
int main() {
  int a=5, b=10, c=-6, d;
  d = a--/2.0 == 2.5 && b/5.0 != 0.0 && c/-3.0 && ++a/2.0 == 2;
  a = b+2 == -2*c != d+a+c;
  printf("d = %d, a = %d\n", d, a);
  return 0;
}
```

```
d = 0, a = 1
```

13. Print the output of the following program for the input `x = 78.467523`. **[3]**

```
#include <stdio.h>
int main (){
  double x, y;
  long int a;
  scanf("%lf", &x);
  y = x - (int)x;
  a = (int)x;
  if (y >= 0.5) ++a;
    printf("x = %0.2lf, y = %0.2lf, a = %ld", x, y, a);
  return 0;
}
```

```
x = 78.47, y = 0.47, a = 78
```

14. Print the output of the following program. **[2]**

```
#include <stdio.h>
int main() {
  int j = (printf("TEN = "), 10);
  printf("%d", j);
  return 0;
}
```

```
TEN = 10
```

— END —

Rough work