INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Stamp / Signature of the Invigilator

EXAMINATION (Mid Semester)	**SEMESTER (Spring 2023-2024)**

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

Subject Number	C	S	1	0	0	0	3	**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**

CS10003 : Programming and Data Structures (Spring 2023-2024)

February 2024 **Mid-Semester Test** Maximum Marks: 60

Instruction to students

- Write your answers in the question paper itself. Be brief and precise. Answer all questions.
- Write the answers only in the respective spaces provided.
- The questions appear on the next six pages.
- Please ask for supplementary sheets during the test if you need more rough space.
- All the questions use the programming language C.
- Not all blanks carry equal marks (unless otherwise stated). In those cases, evaluation will depend on overall correctness.

Do not write anything on this page.

Questions start from the next page.

Q1. What will be the outputs of the following programs? Write in the blank space below each program. **(2×14)**

```
#include <stdio.h>
int main () {
  int a=10, b=10, c=0;
  a = 5 && (b=c);
  printf ("%d %d %d", a, b, c);
  return 0;
}
```

Answer: 0 0 0

```
#include <stdio.h>
int main () {
  int a=10, b=10, c=0;
  a = 5 && (b==c);
  printf ("%d %d %d", a, b, c);
  return 0;
}
```

Answer: 0 10 0

```
#include <stdio.h>
int main () {
  int a=10, b=10, c=9;
  a = (a-b) && (b=c);
  printf ("%d %d %d", a, b, c);
  return 0;
}
```

Answer: 0 10 9

```
#include <stdio.h>
int main () {
  int a=10, b=10, c=9;
  a = (a-b) || (b=c);
  printf ("%d %d %d", a, b, c);
  return 0;
}
```

Answer: 1 9 9

```
#include <stdio.h>
int main () {
  int a=10, b=0;
  char c='0';
  a = (a-b) && (b==c);
  printf ("%d %d %c", a, b, c);
  return 0;
}
```

Answer: 0 0 0

```
#include <stdio.h>
int main () {
  float a = 100/40*10-11.2;
  float b = (float)100/40*10-11.2;
  int c = (int) a;
  printf ("%.2f %.2f %d", a, b, c);
  return 0;
}
```

Answer: 8.80 13.80 8

```
#include <stdio.h>
int main() {
  int i = -2;
  int j = (i>0) - (i<0);
  (j++)?
      printf("%d", --j) :
      printf("%d", ++j);
  return 0;
}
```

Answer: -1

```
#include <stdio.h>
int foo(int n) {
  if (n <= 0) return 0;
  return n + foo(n/2) - foo(n-1);
}
int main() {
  printf ("%d", foo(7));
  return 0;
}
```

Answer: 4

```
#include <stdio.h>
int main () {
  char str[100];
  int d = 'a'-'A', i;
  for (i='A'; i<='Z'; ++i)
    str[i-'A'] = i;
  for (i=1; str[i]!='Z'; ++i)
    if ( (str[i] >= 'A') &&
         (str[i] <= 'Z') )
      str[i] += d;
  for (i=1; str[i]!='Z'; ++i)
    printf ("%c", str[i]);
  return 0;
}
```

Answer: bcdefghijklmnopqrstuvwxy

```
#include <stdio.h>
int main () {
  int S[10], i, n = 10;
  for (i=0; i<n; ++i)
    S[i] = i;
  for (i = 0; i < n/2; ++i) {
    S[i] += S[n-1-i];
    S[n-1-i] = S[i] - S[n-1-i];
    S[i] -= S[n-1-i];
  }
  for (i=0; i<n; ++i)
    printf ("%d", S[i]);
  return 0;
}
```

Answer: 9876543210

```
#include <stdio.h>
int main () {
  char p='I', q='I', r='T';
  if ( (p!=q) || (p>r) )
    printf ("%c %c %c", p, q, r);
  else {
    printf ("%c ", p+2);
    printf ("%c ", q-2);
    printf ("%c ", r-4);
  }
  printf ("%d", r-(p+q)+'A');
  return 0;
}
```

Answer: K G P 3

```
#include <stdio.h>
int main () {
  char choice = 'r';
  switch(choice) {
    case 'b': printf ("B ");
    case 'r': printf ("R ");
    default : printf ("O ");
    case 'g': printf ("G ");
              break;
    case 'y': printf ("Y ");
  }
  return 0;
}
```

Answer: R O G

```
#include <stdio.h>
int count = 0;
int recStat(int n) {
  if (n > 0) {
    count++;
    return recStat(n-1) + count;
  }
  return 0;
}
int main() {
  printf ("%d", recStat(5));
  return 0;
}
```

Answer: 25

```
#include <stdio.h>
#define MAX 150
int main() {
  int one = 1, two = 1, pos = 1;
  while (1) {
    two = one + two;
    one = two - one;
    if (two > MAX)  break;
    pos++;
  }
  printf("%d %d", pos+1, one);
  return 0;
}
```

Answer: 12 144

Q2. In mathematics, a *Niven number* is an integer that is divisible by the sum of its digits. Your task is to complete the following C program which finds whether a given number (within the range $0-999$) is a Niven number or not. **(6)**

```
#include <stdio.h>
int main ( ) {
  unsigned int num, d0, d1, d2, sum;   // Declaration of variables

  /* Take as input a non-negative integer within [0 - 999] */

  scanf ( ____"%u", &num____ );                                    (1)

  /* separate out each digit of the input integer */

  d0 = ____num % 10____ ; // Right digit of the number             (1)

  d1 = ____(num / 10) % 10____ ; // Middle digit of the number     (1)

  d2 = ____num / 100____ ; // Left digit of the number            (1)

  /* Calculate the sum of all digits */

  sum = ____d0 + d1 + d2____ ;                                     (1)

  /* Check the divisibility of the number with the sum of its digits */

  if ( ____num % sum == 0____ )   printf ("YES");                  (1)

  else   printf ("NO");

  return 0;
}
```

Q3. Complete the following recursive C function `recLCM` which finds the *least common multiplier (LCM)* of two numbers a and b (assume $b \geq a$). **(5)**

```
int multiple = 0;   // 'multiple' is a global variable here
int recLCM ( int a, int b ) {

  multiple += ____b____ ;                                          (1)

  if ( ( ____multiple % a == 0____ ) && ( ____multiple % b == 0____ ) )   (1+1)

      return ____multiple____ ;                                    (1)

  else return ____recLCM (a,b)____ ;                               (1)
}
```

Q4. Consider the following C function `MinMaxSort` which takes an array `arr[]` of n integers and arranges the elements in the same array in ascending order. To do so, it finds both the minimum and the maximum elements of the working array `arr[]` (using `findMinIndex` and `findMaxIndex` functions) and swap those two elements to the respective left and right ends of `arr[]` (using `swap` function). It keeps on repeating the same procedure by reducing the working array size having $n, (n-2), (n-4), \ldots$ elements (since both left and right ends of `arr[]` are gradually getting sorted by this process iteratively), until no element is left in `arr[]` to be sorted. Complete the following C functions so that it performs the above mentioned procedure. **(7)**

```
/* Sorting an array with n elements in ascending order */
void MinMaxSort ( int arr[], int n ) {
  int left = 0;   // holds the leftmost index of the working array
  int right = n-1;   // holds the rightmost index of the working array
  int minIndex, maxIndex;

  /* Repeat until the whole array is ordered */

  while ( ____left < right____ ) {                                    (0.5)

      minIndex = findMinIndex ( ____arr, left, right____ );           (0.5)

      swap ( ____arr, left, minIndex____ );                           (0.5)

      maxIndex = findMaxIndex ( ____arr, left, right____ );           (0.5)

      swap ( ____arr, right, maxIndex____ );                          (0.5)

      /* Indicate the working array range and prepare for next iteration */

      left = ____left + 1____ ;   right = ____right - 1____ ;         (1)
  }
}

/* Finding index of the minimum element in the array */
____int____ findMinIndex ( int arr[], int start, int end ) {          (0.5)

  int i, minId = start; // minId will hold the minimum element index in arr[]
  for ( i = start+1; i <= end; ++i )
      /* compare and update for smaller / minimum element index */

      if ( ____arr[minId] > arr[i]____ ) ____minId = i____ ;          (1)

  return minId;
}

/* Finding index of the maximum element in the array */
____int____ findMaxIndex ( int arr[], int start, int end ) {          (0.5)

  int i, maxId = start; // maxId will hold the maximum element index in arr[]
  for ( i = start+1; i <= end; ++i )
      /* compare and update for larger / maximum element index */

      if ( ____arr[maxId] < arr[i]____ ) ____maxId = i____ ;          (1)

  return maxId;
}

/* Swapping arr[i] with arr[j] */
void swap ( int arr[], int i, int j ) {
  int temp = arr[i];
  ____arr[i] = arr[j]____ ;                                           (0.5)
  arr[j] = temp;
}
```

Q5. An array `A[]` stores a permutation of the integers $0, 1, 2, \ldots, n-1$. Choose an index $i_1 \in \{0, 1, 2, \ldots, n-1\}$. If we have `A[`$i_1$`]` = i_2, `A[`i_2`]` = i_3, ..., `A[`i_k`]` = i_1 for some $k \geq 1$, then we say that $(i_1, i_2, i_3, \ldots, i_k)$ is a cycle (of length k) of the permutation. A permutation can always be written as a collection of pairwise disjoint cycles. Your task is to complete the following C program which prints all the cycles of the permutation stored in `A[]`. As an example, let $n = 10$, and consider the following array:

i	:	0	1	2	3	4	5	6	7	8	9
A[i]	:	2	7	8	9	0	5	3	6	4	1

Here, `A[0] = 2, A[2] = 8, A[8] = 4` and `A[4] = 0`, so `(0,2,8,4)` is a cycle. The complete cycle decomposition of this permutation is `(0,2,8,4)(1,7,6,3,9)(5)`.

The program below prints all the cycles one after another in a single line. Each index $i \in \{0, 1, 2, \ldots, n-1\}$ appears in one and exactly one cycle. Therefore, if i is once printed, it will never be printed again. We use an array `printed[]` to store the information about the indices which are printed. The array is initialized to zero. Whenever an i is printed, the corresponding entry in the array is set to a non-zero value. The function stops when all indices are printed. The number of indices printed is stored in the count `nprinted`. **(8)**

```
#include <stdio.h>
int main ()
{
  int A[1000], n, printed[1000], nprinted, i, j;   // Declaration of variables

  /* Assume that the user provides a correct permutation of
  /0,1,2,...,n-1 as input to build the n-element array A[] */
  scanf ("%d", &n);
  for (j = 0; j<n; ++j)   scanf ("%d", &A[j]);

  /* Initialize printed[] array elements to all zeros */

  ____for (j=0; j<n; ++j)    printed[j] = 0;____                    (1)

  nprinted = 0;   // Initialize the number of indices printed
  i = 0;   // Initialize the first index to start with

  /* Repeat until all the indices are printed */

  while ( ____nprinted < n____ ) {                                  (1)

      /* Find the next unprinted index i by searching the array printed[] */

      ____while ( printed[i] )    ++i;____                          (1)

      /* Print the cycle starting from index i */
      printf ("(%d",i);
      printed[i] = ____1____ ;   nprinted = ____nprinted + 1____ ;  (1)

      /* Now let j iterate over all other indices in the current cycle */
      j = A[i];   // Initialize j to the index next to i in the cycle

      while ( ____j != i____ ) { /* Loop on j */                    (1)

          ____printf (",%d",j);____   // Print                      (1)

          /* Update nprinted and an appropriate element in printed[] */

          printed[j] = ____1____ ;   nprinted = ____nprinted + 1____ ;  (1)
```

```
            /* Prepare for the next iteration */

                                    j = A[j];                                    (1)

        }
        printf (")");
    }
    printf ("\n");
    return 0;
}
```

Q6. Complete the following C program which takes as input the number of `rows` (≤ 26) and generates the following character pattern. **(6)**

```
 rows = 2        rows = 3          rows = 4            rows = 5
    A               A                 A                   A
  A B A           A B A             A B A               A B A
                A B C B A         A B C B A           A B C B A
                                A B C D C B A       A B C D C B A
                                                  A B C D E D C B A
```

```
#include <stdio.h>
int main ( ) {
  int i, j, rows;
  char printChar;
  scanf ("%d", &rows);

  for (i=0; i<rows; i++) {

      for (j=0; j<        rows - i        ; j++ )   printf ("  ");          (1)

      printChar = 'A';

      for (j=0;        j<i        ;        j++        ) {                  (1+1)

          printf (" %c", printChar);

          printChar =              printChar + 1              ;            (0.5)
      }

      for (j=i;        j>=0        ;        j--        ) {                  (1+1)

          printf (" %c", printChar);

          printChar =              printChar - 1              ;            (0.5)
      }
      printf ("\n");
  }
  return 0;
}
```

The question paper ends here.