INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Stamp / Signature of the Invigilator

EXAMINATION: End Semester (Autumn 2023-24)	**Answer all (Full marks 100) (Duration: 3 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**	
Marks Obtained											
Question Number	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**Total**
Marks Obtained											

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

Write the answers only in the designated boxes or in the blank lines.

1. Write the output for the following code blocks or provide a description of the error if the code block does not compile: $1 \times 5 = 5$ marks

(a)
```
printf("%d\n", sizeof("x"));
```

Answer: 2

(b)
```
int a, b=110, c=5;
printf("%d\n", (a = b - (c*=5)));
```

Answer: 85

(c)
```
int a, b=110, c=5;
printf("%d\n", (a = b - c*=5));
```

Answer: Error: Since `b-c` is evaluated first which is not an lvalue

(d)
```
int i = 0, j = 1, k = 2, m;
m = i++ || j-- || k%2;
printf("%d %d %d %d", m, i, j, k);
```

Answer: 1 1 0 2 (`j--` will return 1 and then `j` will become 0)

(e)
```
int u=0;
int v = (!u>14) && (u+1);
printf("%d %d\n",u,v);
```

Answer: 0 0 (according to precedence `(!u = 1) > 14` is false)

2. Answer the following questions. $1 \times 5 = 5$ marks

(a) Which one runs faster between the recursive function and the iterative function for finding the GCD of two numbers? **Answer:** iterative

(b) At most how many iterations are needed when linear search is applied on an array containing 2023 distinct elements? **Answer:** 2023

(c) At most how many iterations are needed when binary search is applied on a sorted array containing 2023 distinct elements? **Answer:** 11

(d) Which library contains the function to compute the non-negative square root of a positive floating-point number? **Answer:** `math.h`

(e) Which function of `math.h` is needed to compute the value of $3.14^{20.23}$? **Answer:** `pow` or `powf`

3. Answer the following questions. $2 \times 5 = 10$ marks

(a) Encircle those among the following decimal numbers that have exact values in 4-byte floating-point representation.

-2.65625 $\quad -1.025$ $\quad 0.525$ $\quad 1.625$ $\quad 2.4375$

Answer: $-2.65625,\ 1.625,\ 2.4375$

(b) Encircle those among the following expressions that can be evaluated in `main()` without using any loop or recursive function. The real number $x > 0$ and the integer $n > 0$ are given as input.

$$\sum_{i=0}^{n} x^i \qquad \sum_{i=0}^{n} i \cdot x^i \qquad \sum_{i=0}^{n} \frac{i}{x^i} \qquad \sum_{i=0}^{n} x^{\frac{1}{i}} \qquad \log_2\left(\sum_{i=1}^{n} i \cdot x^i\right)$$

Answer: All excepting the 4th

(c) What is the value of (`'z'-'a'`) + (`'z'-'b'`) + $\cdots$ + (`'z'-'z'`)? **Answer:** $25 \times 26/2 = 325$

(d) Consider the following code block:

```
typedef struct{
  int x, y;
}point;
typedef struct{
  point vertex[3];
}triangle;
triangle tSet[100];
```

What is the total space (in bytes) required for the array `tSet`? **Answer:** 2400

(e) Recall that Fibonacci numbers are recursively defined as:

$$f(0) = 0,\ f(1) = 1,\ f(n) = f(n-1) + f(n-2)\ \forall\ n \geq 2.$$

How many recursive function calls occur to compute $f(5)$? **Answer:** 7

4. Write the output of the following program: $1+1+3 = 5$ marks

```
#include<stdio.h>

int main(){
  char *arr[] = {"ant", "bat", "cat", "dog", "egg", "fly"};
  char *ptr1, *ptr2;
  int i;
  unsigned int k;
  ptr1 = (arr+1)[2];
  ptr2 = (*arr+2);

  for(i=0; i<10; i++){
    k = k + (ptr2 - ptr1);
    k = k%4;
  }
  printf("Out1: %s || Out2: %s || Out3: %s\n", ptr1, ptr2, arr[k]);

  return 0;
}
```

Answer: `Out1: dog || Out2: t || Out3: ant` **Errata: initially, k = 0**

5. *Outliers* in an array are those of its elements that lie outside the interval $[\mu - \sigma, \mu + \sigma]$, where μ and σ are the respective mean and standard deviation of all its elements. Fill in the blanks in the following code block for computing the outliers in a given array. The functions must allocate memory dynamically in the right amount. $5 \times 2 = 10$ marks

```
//Compute mean & standard deviation and return/pass their values
double mean_std_dev (double *arr, int n, double *retmean) {
  double sum = 0.0;
  for (int i = 0; i < n; i++)
    sum += arr[i];
  *retmean= sum / n;
  for (sum = 0.0, i = 0; i < n; i++)

    sum += _____________________________________________; // 2 marks

  return sqrt(sum / n);
}

//Find elements outside the desired range
int outliers (double *arr, int n, double **retout) {
  int k=0, j=0;
  double m, s; //mean and std. dev.

  s = _____________________________________________; // 2 marks
  for (int i = 0; i < n; i++) {
    if (arr[i] < m-s || arr[i] > m+s)
      k++;
  }
  *retout = _____________________________________________; // 2 marks
  for (int i = 0; i < n; i++) {
    if (arr [i] < m-s || arr [i] > m+s)

      _____________________________________________; // 2 marks
  }
  return k;
}

int main() {
  int n, m, i;
  double *a, *b;
  printf("Number of data: ");
  scanf("%d", &n);
  a = (double *)malloc(sizeof(double)*n);
  for(i=0; i<n; i++)
    scanf("%lf",&a[i]);

  // get outliers //
  m = _____________________________________________; // 2 marks
  printf("Outliers are: ");
  for(i = 0; i < m; i++)
    printf("%lf ",b[i]);
  return 1;
}
```

ANSWER

```
sum += pow (arr[i] - *retmean, 2);
s = mean_std_dev(arr, n, &m);
*retout = (double *)malloc(l*sizeof(double));
(*retout)[j++] = arr[i];
m = outliers(a,n,&b);
```

6. A *double right-shift* on a digit means increasing its value by 2 with the "wrap-around effect". That is, the digits $0,1,2,\ldots,7,8,9$ are mapped to $2,3,4,\ldots,9,0,1$, respectively. Fill in the blanks for the following C program that counts the number of digits in a given string, applies a double right-shift on each digit by calling a function, and finally prints the modified string. For example, **Prog1594** is modified to **Prog3716**. 7 marks

```
#include <stdio.h>

int digitFun (char *s); // function prototype

int main(){
  char str[20];
  printf("\nEnter the string: ");

  scanf("___________", str);                              // 1 mark

  printf("No. of digits: _____", digitFun(__________));   // 2 marks

  printf("\nString after change: %s", str);

  return 0;
}

int digitFun (char *s){
  int n = 0;
  while (*s){
    if (*s >= '0' && *s <= ___________) {                 // 1 mark

      n = ____________;                                   // 1 mark

      *s = ((*s - _________) + 2) % __________ + '0';     // 2 marks
    }
    s++;
  }
  return n;
}
```

ANSWER

```
scanf("%s", str);
printf("No. of digits: %d", digitFun(str));
if (*s >= '0' && *s <= '9')
n = n + 1;
*s = ((*s - '0') + 2) % 10 + '0';
```

7. Suppose in a C program the three strings **`IIT`**, **`Kharagpur`**, and **`Programming`** have to be stored. Calculate the memory requirement in bytes for each of the following two options of storing the strings in a computer with 64-bit addresses: $2+3=5$ marks

   (a) Option-1: a statically declared 2D array of characters with 10 rows and 20 columns, with each word starting at a new row of the 2D array. **Answer:** $10 \times 20 = 200$ bytes

   (b) Option-2: a dynamically declared array of length three, where each of the three array entries is a character pointer containing the starting address of one of the three strings. **Answer:** $3 \times 8 + 4 + 10 + 12 = 50$ bytes

8. Fill in the blanks for the following C program that reads in the information about a group of students from the user, and then prints the longest name among the group. 5 marks

```
#include <stdio.h>
#include <string.h>
#define CLASS_SIZE 10
#define MAX_LEN 20

typedef struct{
  char name[MAX_LEN];
  int age;
  float height; // in cm
}student;

_______ findlongestname(_____________________ class[]){ // 2 marks

  int i, maxlen = 0, currlen, index;

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

    if ((currlen = strlen(_____________________)) > maxlen){
      maxlen = currlen;                                  // 1 mark

      index = ___________;                               // 1 mark
    }
  }
  return _________________;                              // 1 mark
}

int main(){
  student class[CLASS_SIZE];
  int i, index;

  for (i = 0; i < CLASS_SIZE; i++){
    printf("\nEnter name: ");
    scanf(" %[^\n]", class[i].name);
    printf("\nEnter age: ");
    scanf(" %d", &class[i].age);
    printf("\nEnter height: ");
    scanf(" %f", &class[i].height);
  }

  index = findlongestname(class);
  printf("\nThe longest name is: %s", class[index].name);

  return 0;
}
```

ANSWER

```
int findlongestname (student class[])
if ((currlen = strlen(class[i].name)) > maxlen)
index = i;
return index;
```

9. Fill in the blanks in the following C program that uses structures to implement set and the operation of intersection between two sets. Here a set consists of at most 100 two-dimensional points with integer coordinates. For example, for the two sets $A = \{(-2,3),(1,-2),(4,3),(5,7)\}$ and $B = \{(1,-2),(2,2),(3,-7),(4,3),(4,6)\}$, we get $A \cap B = \{(1,-2),(4,3)\}$. [10 marks]

```
#include <stdio.h>

typedef struct{
  int x, y;
}point;

typedef struct{
  point p[20];
}Set;

// Scan the elements of a Set
Set read(int s){
  Set S;
  for(int i=0; i<s; i++)
    scanf(_____________________________________________); // 1 mark

  return _____________; // 1 mark
}

// Display the elements of a Set
void print(Set S, int s){
  for(int i=0; i<s; i++)
    printf(_____________________________________________); // 1 mark
}

// Compute the intersection of two Sets
Set findIn(Set R, Set S, int r, int s, int *u){
  Set t;
  int i, j, m=0;
  for(i=0; i<r; i++){
    for(j=0; j<s; j++){

      if(_____________________________________________){ // 1 mark

        _____________________________________;          // 1 mark

        _____________________________________;          // 1 mark
        m++;
      }
    }
  }
  *u = _____________________;                           // 1 mark
  return t;
}
```

```
int main(){
  Set A, B, C;
  int m, n, k;

  printf("Enter the number of elements in set A:\n");
  scanf("%d", &m);
  printf("Enter the elements in set A:\n");
  A = read(m);
  printf("Enter the number of elements in set B:\n");
  scanf("%d", &n);
  printf("Enter the elements in set B:\n");
  B = read(n);

  C = findIn(_____); // 1 mark

  if(_____)          // 1 mark
    printf("No intersection\n");
  else{
    printf("Intersection: ");

    print(_____);          // 1 mark
  }
  return 0;
}
```

ANSWER

```
scanf("%d%d", &S.p[i].x, &S.p[i].y);
return S;
printf("(%d, %d)", S.p[i].x, S.p[i].y);
if((R.p[i].x == S.p[j].x) && (R.p[i].y == S.p[j].y))
t.p[m].x = R.p[i].x
t.p[m].y = R.p[i].y;
*u = m;
C = findIn(A, B, m, n, &k);
if(k == 0)
print(C, k);
```

10. Consider the following code. Given that the first line prints **`A = 2796`**, write what each other line prints (as a comment in the adjacent blank line). $1 \times 3 = 3$ marks

```
#include <stdio.h>

int main(){
  int A[20] = {2, 3, 4, 5, 6, 7, 8, 9, 8, 7 };

  printf("A      = %u\n", A); // A = 2796

  printf("A + 1  = %u\n", A + 1); //_____________________________

  printf("&A     = %u\n", &A); //_____________________________

  printf("&A + 1 = %u\n", &A + 1); //_____________________________

  return 0;
}
```

ANSWER

```
A + 1  = 2800
&A     = 2796
&A + 1 = 2876
```

11. Fill in the blanks in the following code block. Its objective is to find out whether all elements of an array **c[]**, taking into account their repetitions if any, are present in an array **b[]**. For example, all elements of $\{-4,7,7\}$ are not present in $\{-2,9,-4,7,9\}$ but present in $\{2,9,7,-4,7,9\}$ indeed. Note that the function **linSearch** takes as input an array, its number of elements, and a key element to be searched; it returns the array index when a match is found, and returns -1 otherwise. [10 marks]

```
#include <stdio.h>

int linSearch(int a[], int n, int x){
  for(int i=0; i<n; i++)
    if(x==a[i])
      return i;
  return -1;
}

int main(){
  int b[20] = {2, 4, 3, 7, 9, 9, 19, 7, 67, 56, 4, 5}, nb = 12;
  int c[20] = {4, 8, 7, -9, 7}, nc = 5;

  int i, idx, count = 0, n0 = ____________;                  // 1 mark

  for(i = 0; ____________________; i ++){                    // 2 marks

    idx = linSearch(c, _____________, b[_____________]);    // 2 marks
    if(idx != -1){
      count++;
      c[idx] = c[_____________________];                    // 2 marks

      nc = _________________;                               // 1 mark
    }
  }
  if(count == _________________)                            // 1 mark
    printf("All elements there\n");
  else
    printf("%d elements missing\n", __________ - count); // 1 mark

  return 0;
}
```

ANSWER

```
n0 = nc;
i < nb;
idx = linSearch(c, nc, b[i]);
c[idx] = c[nc-1];
nc = nc-1;
if(count == n0) // correction: nc is replaced by n0
n0
```

12. Let r be the last digit of your roll number. For example, if your roll number is 23YZ10094, then $r = 4$. Consider the array $\boxed{33+r \mid 21+r \mid 81+r \mid 56+r \mid 12+r \mid 45+r \mid 68+r}$. How many swaps are done when Bubble Sort is applied on it to arrange its elements in the increasing order?

Answer: 9 [5 marks]

13. The function **spiralTraversal** prints the elements of a two-dimensional array **a[r][c]** in a spiral fashion. Fill up its blanks. Note that it works for both **r = c** and **r ≠ c**. For example, for the left array given below, the spiral traversal is **1 2 3 4 8 6 2 3 4 5 9 5 6 7 7 8**, and for the right, it is **1 2 3 4 8 6 7 8 9 5 6 7**. 10 marks

```
1 2 3 4          1 2 3 4
5 6 7 8          5 6 7 8
9 8 7 6          9 8 7 6
5 4 3 2
```

```
void spiralTraversal(int r, int c, int a[r][c]){
  int top = 0, bottom = r - 1, left = 0, right = c - 1;
  int dir = 0; // 0 = right, 1 = down, 2 = left, 3 = up

  while(top <= ____________________ && left <= ____________________){
                                                          // 2 marks
    if(dir == 0){
      for(int i = left; i <= right; i++){
          printf("%d ", a[top][i]);
      }
      top = ____________________;                         // 1 mark
    }
    else if(dir == 1){
      for(int i = top; i <= bottom; i++){
        printf("%d ", a[____________][____________]);     // 2 marks
      }
      right = ____________________;                       // 1 mark
    }
    else if(dir == 2){
      for(int i = right; i >= left; i--){
        printf("%d ", a[bottom][i]);
      }
      bottom = ____________________;                      // 1 mark
    }
    else if(dir == 3){
      for(int i = ________________; i >= top; i--){        // 1 mark
        printf("%d ", a[i][left]);
      }
      left = ____________________;                        // 1 mark
    }
    dir = (dir + 1) % ________________; // Change direction // 1 mark

  } //end while
}
```

ANSWER

```
while(top <= bottom && left <=right)
top + 1
a[i][right]
right - 1
bottom - 1
left + 1
for(int i = bottom; i >= top; i--)
4
```

14. Given a positive integer n as input, the following code finds in how many ways n can be written as a sum of the two summands 1 and 2 such that the sum has an **odd number of summands**. For example, for $n = 3$, there is only one way $(1+1+1)$; for $n = 4$, there are only three ways $(1+1+2, 1+2+1, 2+1+1)$; for $n = 5$, there are only four ways $(1+1+1+1+1, 1+2+2, 2+1+2, 2+2+1)$. Fill up the blanks.
What will be the answer for $n = 7$? **Answer:** 11 $8+2=10$ marks

```
#include <stdio.h>
int ways(int n){

  if(_____________________________) return 0;                    // 1 mark

  if(n<=3) return _____________________________;                 // 2 marks

  return ____________________________________________________;   // 5 marks
}

int main(){
  int n;
  printf("Enter n: "); scanf("%d", &n);
  printf("#ways = %d\n", ways(n));
  return 0;
}
```

ANSWER

```
if(n==0) return 0;
if(n<=3) return 1;

return ways(n-2) + 2*ways(n-3) + ways(n-4);
or
return ((n%3==0)?(ways(n-1)+ways(n-2)-1):
       ((n%3==1)?(ways(n-1)+ways(n-2)+1):
       (ways(n-1)+ways(n-2))))
or
return (ways(n-1)+ways(n-2)+(2*n%3)-1)

ways(n = 7) = 11
```

— END —

Space for rough work

Space for rough work