INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Stamp / Signature of the Invigilator

EXAMINATION: End Semester (Autumn 2024-25)	**Answer all (Total marks: 90, Duration: 3 hours)**

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 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 No.:	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
Marks Obtained											
Question No.:	**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**

1. Answer all questions. The end of this question paper is marked by —— **END** ——
2. Write your answers in the specified **blanks**, using **pen only**.
3. You may do **Rough Work** in the designated areas or any white space, as long as it does not encroach or interfere with your answers.
4. If you use **Extra Sheets** for rough works, do not submit them.

1. Write the **correct option** in the table below. **$1 \times 10 = 10$ marks**

i	ii	iii	iv	v	vi	vii	viii	ix	x

i) Which of the following is not a function of `math.h`?

(A) `log` **(B)** `min` **(C)** `tan` **(D)** `atan` **(E)** `pow`

ii) For which of the following problems would even the most efficient code run the slowest?

(A) Sorting **(B)** Binary search **(C)** Tower of Hanoi **(D)** GCD **(E)** Fibonacci sequence

iii) Which one of the following is most accurate to define π using `#define`?

(A) `22.0/7.0` **(B)** `2.0 * asin(1.0)` **(C)** `0.5 * asin(1.0)` **(D)** `3.14` **(E)** `3.14159`

iv) Which one of the following is an incorrect way to declare and initialize a string?

(A) `char t[] = "IIT";` **(B)** `char t[] = "IIT\0";` **(C)** `char t[4] = "IIT\0";`
(D) `char t[4] = "IIT";` **(E)** `char t[] = {'I','I','T'};`

v) How many comparisons will be required by the best-known algorithm to find both the largest and the smallest elements in array with 1024 integers?

(A) 2047 **(B)** 2046 **(C)** 1534 **(D)** 1034 **(E)** 1033

vi) Which one of the following will give a compilation error?

(A) `int *p = 5;` **(B)** `int ***q;` **(C)** `int *a[5];`
(D) `int **b[5];` **(E)** `int **c[1][1];`

vii) Select the output of `printf("%d\n", ('A'-'b'=='B'-'c') + ('D'-'1'=='E'-'2'));`

(A) 1 **(B)** 2 **(C)** 129 **(D)** 130 **(E)** 133

viii) Select the output of `printf("%d", (19/2) & (47>>4));`

(A) `10` (Correct = 0) **(B)** `11` **(C)** `20` **(D)** `21` **(E)** `22`

Note: This question doesn't contain the correct answer and hence will be disregarded.

ix) Select the output of `printf("%x\n", (1<<16) + 15);`

(A) `1000f` **(B)** `101f0` **(C)** `fffff` **(D)** `f000f` **(E)** `ffff0`

x) Select the output of `printf("/*Shh!*/%s%c", "%c", "%c"[0]);`

(A) `/*Shh!*/%c%` **(B)** `%c%` **(C)** `/*Shh!*/cc` **(D)** `cc` **(E)** Compilation error

Space for rough work

2. Write the **correct option** in the table below. **$3 \times 10 = 30$ marks**

i	ii	iii	iv	v	vi	vii	viii	ix	x

i) For a 2D array `int a[4][5]`, if the address of `a[0][0]` is `0x33b0`, then what will be the addresses of `a[1][0]`, `a[2][0]`, and `a[3][4]`? (An integer needs 4 bytes.)

(A) `0x33c0`, `0x33d4`, `0x33f8` **(B)** `0x33c4`, `0x33d8`, `0x33f8` **(C)** `0x33c0`, `0x33d8`, `0x33fc` **(D)** `0x33c4`, `0x33d8`, `0x33fc` **(E)** `0x33c4`, `0x33d4`, `0x33fc`

ii) Suppose that the array `a[]` contains five distinct positive integers. Which of its elements will be stored in `x` when the following loop terminates?

```
int i, x = 0, y = 0, z = 0;
for (i = 0; i < 5; i++) {
    if (a[i] > z) {
        x = y; y = z; z = a[i];
    } else if (a[i] > y) {
        x = y; y = a[i];
    } else if (a[i] > x) {
        x = a[i];
    }
}
```

(A) Largest element **(B)** 2nd largest element **(C)** 3rd largest element **(D)** 4th largest element **(E)** Smallest element

iii) Suppose that A, B, C are three matrices, each of size 3×3, implemented as 2D arrays, which are `a[][]`, `b[][]`, and `c[][]`, respectively, and intialized as:

$$\texttt{a[i][j]} = \begin{cases} 1 & \text{if } \texttt{i} = \texttt{j} \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad \texttt{b[i][j]} = \texttt{c[i][j]} = 0 \text{ for } 0 \le \texttt{i} \le 3,\ 0 \le \texttt{j} \le 3.$$

What will be correct in the context of matrix operations after executing the following code?

```
for (int i = 0; i < 3; i++)
  for (int j = 0; j < 3; j++)
    for (int k = 0; k < 3; k++)
      b[i][j] += a[i][k] * a[k][j];

for (int i = 0; i < 3; i++)
  for (int j = 0; j < 3; j++)
    for (int k = 0; k < 3; k++)
      c[i][j] += b[i][k] * b[k][j];
```

(A) $C = A$ **(B)** $C \neq A$ **(C)** $C \neq B$ **(D)** $B \neq A$ **(E)** $A + B = C$

iv) The following 32 bits represent a floating-point number in IEEE 754 single-precision format: `11000000 11000000 00000000 00000000`
What is its real value in decimal number system?

(A) -2.000 (Correct $= -6.0$) **(B)** $+2.000$ **(C)** -4.500 **(D)** $+4.500$ **(E)** $+2.750$

Note: This question doesn't contain the correct answer and hence will be disregarded.

v) What will be the binary value of the decimal number 0.4, expressed within 20 bits after the decimal point?

(A) `0.011000100100011001110` **(B)** `0.01100110010011100 1101`
(C) `0.01100111010111001110` **(D)** `0.01100110011001100`
(E) `0.01010110011011100`

vi) The following 2D array of size 8×2 contains (x, y) coordinates of eight points:

```
{{2, 13}, {1, 11}, {1, 18}, {2, 18}, {2, 16}, {2, 15}, {1, 16}, {1, 13}}
```

What will be the sorted points if Merge Sort is applied on this array? Assume that the sorting is done using x-coordinates only.

(A) `{{1, 11}, {1, 13}, {1, 16}, {1, 18}, {2, 13}, {2, 18}, {2, 16}, {2, 15}}`
(B) `{{1, 11}, {1, 13}, {1, 16}, {1, 18}, {2, 13}, {2, 15}, {2, 16}, {2, 18}}`
(C) `{{1, 11}, {1, 16}, {1, 13}, {1, 18}, {2, 13}, {2, 15}, {2, 16}, {2, 18}}`
(D) `{{1, 11}, {2, 13}, {1, 18}, {2, 18}, {1, 13}, {1, 16}, {2, 15}, {2, 16}}`
(E) `{{1, 11}, {1, 18}, {1, 16}, {1, 13}, {2, 13}, {2, 18}, {2, 16}, {2, 15}}`

vii) A function $f(n)$ on non-negative integers is 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 $f(10)$?

(A) 31 **(B)** 176 **(C)** $2^{10} - 1$ **(D)** 2^{10} **(E)** $2^{10} + 1$

viii) The strings `IIT`, `Kharagpur`, and `India` are stored in a statically declared 2D array of characters with 5 rows and 20 columns, with each word starting at a new row.

The strings `PDS`, `Exam`, and `2024` are stored in 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.

Given that the computer has 64-bit addresses, what is the total memory requirement in bytes for storing these six strings? There is some unused space in the 1st array, which should also be taken into account in your calculations.

(A) 126 **(B)** 131 **(C)** 134 **(D)** 138 **(E)** 141

ix) Consider the array

23	11	71	46	2	35	58

. How many swaps are done when Bubble Sort is applied on it to arrange its elements in the increasing order?

(A) 8 **(B)** 9 **(C)** 10 **(D)** 11 **(E)** 12

x) How many bytes are required to allocate the space for `wheel w[8]`, with the following structures? Assume that the computer allocates space just as much as needed, without any padding.

```
typedef struct point{
  float x, y;};
typedef struct line{
  point p, q;};
typedef struct circle{
  point c; floar r;};
typedef struct wheel{
  circle in, out; line spoke[8];};
```

(A) 152 **(B)** 960 **(C)** 1216 **(D)** 1280 **(E)** 1408

Space for rough work

3. The following code computes and prints all possible ways of distributing `m` identical toffees among `n` distinguishable children such that each child receives at least one toffee. For example, 5 toffees can be distributed among 3 children in 6 possible ways: `{1, 1, 3}`, `{1, 2, 2}`, `{1, 3, 1}`, `{2, 1, 2}`, `{2, 2, 1}`, `{3, 1, 1}`.

Fill in the blanks. **$7\frac{1}{2}$ marks**

```
#include<stdio.h>

int dist(int a[], int k, int n, int m){
   int num = 0, i;

   if(n == 1){
     a[k] = _____________________________; // 2 marks

     return _____________________________; // 1 mark
   }

   for(i = 1; i <= m-n+1; i++){

     a[k] = _____________________________; // 2 marks

     num += _____________________________; // 2.5 marks
   }
   return num;
}

int main(){
  int a[10], n, m; // n < 11, m > n-1
  printf("Enter number of children and number of toffees: ");
  scanf("%d%d", &n, &m);
  printf("Number of distributions is %d\n", dist(A, 0, n, m));
  return 0;
}
```

ANSWER

```
a[k] = m;
return 1;
a[k] = i;
num += dist(A, k+1, n-1, m-i);
```

Space for rough work

4. An array `a[]` contains `n` cells, each with a positive integer. For each cell `a[i]`, its element represents the maximum length of a jump that can be made forward from that cell to reach another cell. The task is to find the minimum number of jumps to reach the last cell, `a[n-1]`, starting from the first cell, `a[0]`.

For example, if `a[] = {3, 1, 3, 1, 4, 2, 1, 1, 2}`, the minimum number of jumps required is three:

- First, jump from `a[0]` to `a[2]`.
- Then, jump from `a[2]` to `a[4]`.
- Finally, jump from `a[4]` to the last cell, `a[8]`.

Here, `a[0]` contains 3, so it allows a jump to any of `a[1]`, `a[2]`, or `a[3]`. We jumped to `a[2]` because it leads to the minimum number of total jumps.

Fill in the blanks in the following code to perform this task. **$1\frac{1}{2} \times 5 = 7\frac{1}{2}$ marks**

```
#include<stdio.h>

int minJumps(int a[], int n){
  int opt[n], i, j, k, min; // opt[i] = min #jumps from a[i] to a[n]

  opt[n-1] = _________________________;

  for(i = n-2; i >= 0; i--){

    min = _________________________;

    k = (_________________________) ? a[i] : n-i-1;

    for(j = 2; j <= k; j++){

      if (min > 1 + opt[i+j])

        min = ____________________;

    opt[i] = min;
  }

  return _________________________;
}

int main() {
  // int a[] = {3, 1, 3, 1, 4, 2, 1, 1, 2}; // answer = 3
  int a[] = {3, 1, 3, 1, 4, 2, 1, 7, 2, 1, 1, 3, 1, 1}; // answer = 4
  int n = sizeof(a) / sizeof(a[0]);
  printf("Minimum jumps to reach the end: %d\n", minJumps(a, n));
  return 0;
}
```

ANSWER

```
opt[n-1] = 0;
min = 1 + opt[i+1];
k = (a[i] < n-i-1) ? a[i] : n-i-1;
min = 1 + opt[i+j];
return opt[0];
```

Space for rough work

Space for rough work

5. An array of integers is considered *equitable* if two conditions are met:
– All distinct elements occur the same number of times.
– When sorted, these distinct elements form a consecutive sequence.
For example, `{7,5,6,5,7,6}` is equitable because each distinct element appears exactly twice, and when sorted, the distinct elements form the consecutive sequence `{5,6,7}`.
Given an input array of 10 integers, the following code prints `YES` if the array is equitable, and `NO` otherwise. Fill in the blanks. **$2 \times 5 = 10$ marks**

```
#include<stdio.h>
int main(){
  int a[10], freq[10], n=10, i=0, j, min, fmin;

  for(i=0; i<n; i++)
    scanf("%d", &a[i]);

  min = a[0];
  for(i=1; i<n; i++)
    if(a[i]<min)
      min = a[i];

  for(fmin=0, i=0; i<n; i++){

    if(_____________________________________)
      fmin++;
  }

  int m = _____________________________________;

  for(i=0; i<m; i++)
    freq[i] = 0;

  for(i=0; i<n; i++){
    j = a[i]-min;

    if (_________________ || ___________________){

      printf("NO\n");
      return 1;
    }
    freq[j]++;
  }

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

    if (_____________________________________){
      printf("NO\n");
      return 1;
    }
  }

  printf("YES\n");
  return 1;
}
```

ANSWER

```
a[i] == min
int m = n/fmin;
if (j<0 || j>m-1)
if (freq[i] != fmin)
```

Space for rough work

6. Recall that Fibonacci numbers are recursively defined as:

$$f(0) = 0,\ f(1) = 1,\ f(n) = f(n-1) + f(n-2) \text{ if } n \geq 2.$$

For the sequence $g(n) = 2f(n)+3\ \forall\ n \geq 0$, the following code computes its value with $n\ (> 0)$ as input. Fill in the blanks. **5 marks**

```
#include <stdio.h>

int g(int n){

  if (n == 0) return _________________; // 1 mark

  if (n == 1) return _________________; // 1 mark


  return _____________________________; // 3 marks
}

int main(){
  int n, i;
  printf("Enter n: ");
  scanf("%d", &n);
  for(i=0; i<=n; i++)
    printf("g(%d) = %d\n", i, g(i));
  return 0;
}
```

ANSWER

```
if (n == 0) return 3;
if (n == 1) return 5;
return g(n-1) + g(n-2) - 3;
```

Space for rough work

7. Write the answers.

i) Given a queue `Q` with 5 integer elements (from front to back: `1, 3, 5, 7, 9`), and an empty stack `S`, remove the elements one-by-one from `Q` and insert them into `S`, then remove them one-by-one from `S` and re-insert them into `Q`. Write the elements of `Q` from front to back.

Answer: 9, 7, 5, 3, 1

$2\frac{1}{2}$ marks

ii) The following operations are performed on a queue `Q` in this order:
Enqueue `1`; Enqueue `9`; Enqueue `4`; Dequeue; Dequeue; Enqueue `7`; Enqueue `0`; Dequeue; Enqueue `8`; Enqueue `1`; Enqueue `5`; Dequeue.
Which will be the smallest and the largest elements of `Q`? Assume that `Q` is initially empty and capable of holding 100 integers.

Answer: 0, 8

$2\frac{1}{2}$ marks

Space for rough work

8. The following code inserts integer elements in a linked list (initially empty) in increasing order, discarding duplicate elements, stopping by 0. Fill in the blanks. **$1 \times 15 = 15$ marks**

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

typedef struct mynode{
  int x;

  ____________________________________________ *next;

} node;

node *insert_linked_list(node *head, int x){
  node *p, *q, *new;

  new = ____________________________________________; // new node for x

  new->x = _________________________;

  new->next = _____________________;

  if (head == NULL) // Linked list is empty

    return _________________________;

  p = q = _________________________;

  // Case 1: x < some element of linked list
  while(p != NULL){
    if (x == p->x)  // It's a duplicate element, so don't insert.

      return _____________________;

    else if (x < p->x){ // Insert new before p.
      if (p == head){
        new->next = head;

        return _____________________;
      }

      new->next = _________________;

      q->next = ___________________;

      return _____________________;
    }
    else{
      q = _________________________;

      p = _________________________;
    }
```

```
  } // end while

  // Case 2: x > all elements of linked list

  q->next = ________________________;

  return ____________________________;
}

int main(){
  int k;
  node *head = NULL;
  printf("Enter nonzero elements and 0 to stop: ");
  do{
     scanf("%d", &k);
     if(k != 0) head = insert_linked_list(head, k);
     else break;
  }while(1);

  return 1;
}
```

ANSWER

Full code given below.

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

typedef struct mynode{
  int x;
  struct mynode *next;
} node;

node *insert_linked_list(node *head, int x){
  node *p, *q, *new;
  new = (node *)malloc(sizeof(node)); // new node for x
  new->x = x, new->next = NULL;

  if (head == NULL) // Linked list is empty
    return new;

  p = q = head; // Once p moves forward, q will be the node before p.

  // Case 1: x < some element of linked list
  while(p != NULL){
    if (x == p->x)  // It's a duplicate element, so don't insert.
      return head;
    else if (x < p->x){ // Insert new before p.
      if (p == head){
        new->next = head;
        return new;
      }
      new->next = p;  // Insert before p
```

```
29        q->next = new;
30        return head;
31      }
32      else{
33        q = p;
34        p = p->next; // q is the node before p.
35      }
36    }
37
38    // Case 2: x > all elements of linked list
39    q->next = new;
40    return head;
41  }
42
43  int main(){
44    int k;
45    node *head = NULL;
46    printf("Enter nonzero elements and 0 to stop: ");
47    do{
48       scanf("%d", &k);
49       if(k != 0) head = insert_linked_list(head, k);
50       else break;
51    }while(1);
52
53    return 1;
54  }
```

END