**General Instructions:**

* **Follow the instructions given in each section.**
* **Make sure that you attempt the questions in order.**

**SECTION-A (10\*1 mark=10 marks)**

***(All questions are compulsory)***

Q1 1. What is the purpose of the rewind() function in C file handling?

a) to close a file

b) to move the file pointer to the beginning of the file

c) to move the file pointer to the end of the file

d) to delete the contents of the file

Answer: b) to move the file pointer to the beginning of the file

Q2 1. What is the purpose of the fflush() function in C file handling?

a) to close a file

b) to clear the buffer for a file stream

c) to write data to a file

d) to read data from a file

Answer: b) to clear the buffer for a file stream

Q3 1. What is the syntax for closing a file in C?

a) close(file);

b) fclose(file);

c) close\_file(file);

d) fclose(file);

Answer: b) fclose(file);

Q4 1. What is the syntax for accessing a member of a union in C?

a. union\_name.member\_name

b. union\_name->member\_name

c. union\_name[member\_name]

d. None of the above

Answer: a

Q5 1. What is the purpose of the sizeof operator in C?

a. To determine the size of a data type or variable in bytes

b. To declare a new variable

c. To define a new data type

d. None of the above

Answer: a

Q6 1. Which of the following statements is true about unions in C?

a. Only one member of a union can be used at a time

b. All members of a union can be used at the same time

c. Union members can be of different sizes

d. Union members must be of the same size

Answer: a

Q7 1. Which of the following is an example of an enumeration in C?

A) enum { RED, GREEN, BLUE };

B) enum { 1, 2, 3 };

C) enum { "Monday", "Tuesday", "Wednesday" };

D) None of the above

Answer: A) enum { RED, GREEN, BLUE};

Q8 1. What is the keyword used to define a bit field in a struct in C?

A) bit

B) int

C) char

D) typedef

Answer: B) int

Q9 1. Which of the following is not required in a recursive function?

a) Base case

b) Recursive call

c) Loop

d) All of the above are required

Answer: c) Loop

Q10 1. What is the time complexity of a recursive function that calls itself n times?

a) O(n)

b) O(log n)

c) O(n log n)

d) O(n^2)

Answer: a) O(n)

**SECTION-B (5\*2 mark=10 marks)**

***(All questions are compulsory)***

Q11 What is the output of the following recursive function when called with an argument of 6?

int sumDigits(int n) {

if (n == 0) {

return 0;

}

return n%10 + sumDigits(n/10);

}

A. 6

B. 15

C. 21

D. 30

Q12 What is the output of the following code?

#include <stdio.h>

struct point {

int x;

int y;

};

void printPoint(struct point p) {

printf("(%d,%d)\n", p.x, p.y);

}

int main() {

struct point p1 = { 5, 10 };

struct point p2 = { 15, 20 };

printPoint(p1);

printPoint(p2);

return 0;

}

A. (5,10) (15,20)

B. (15,20) (5,10)

C. (5,20) (15,10)

D. Error

Correct answer: A

Q13 What is the correct syntax for declaring a pointer to a student structure called s\_ptr?

a. student \*s\_ptr;

b. struct \*s\_ptr student;

c. \*student s\_ptr;

d. struct student \*\*s\_ptr;

Correct answer: a

Q14 What is the output of the following code snippet?

#include <stdio.h>

enum numbers { ONE, TWO, THREE };

int main() {

enum numbers n = TWO;

switch(n) {

case ONE: printf("1"); break;

case TWO: printf("2"); break;

case THREE: printf("3"); break;

}

return 0;

}

a) 1

b) 2

c) 3

d) Compiler error

Answer: b

Q15 What is the output of the following program?

#include <stdio.h>

struct point {

int x;

int y;

};

int main() {

struct point points[3] = {{1, 2}, {3, 4}, {5, 6}};

for (int i = 0; i < 3; i++) {

printf("Point %d: (%d, %d)\n", i+1, points[i].x, points[i].y);

}

return 0;

}

a) The program prints the coordinates of 3 points on the screen: (1, 2), (3, 4), and (5, 6).

b) The program produces an error during compilation.

c) The program compiles successfully but does not produce any output.

d) The program produces a runtime error.

Correct answer: a

**SECTION-C(Coding Question) (2x5 marks=5 marks)**

Q16. Imagine that you are at the base of a staircase that has several steps, numbered from 1 to n. You want to climb to the top of the staircase, but each step has a cost associated with it. You are given an array called "cost" that contains the cost of each step. Specifically, cost[i] represents the cost of climbing the ith step on the staircase.

You have two options for climbing each step. You can either pay the cost of the step and climb one step, or pay the cost of the step and climb two steps. Once you have climbed a step, you cannot go back down.

Your goal is to find the minimum cost required to reach the top of the staircase, which is the nth step

**Input:**

One line with integer **n** giving the size of the array.

**n** integers separated giving the values cost[i].

**Constraints:**

2 <= cost.length <= 100

0 <= cost[i] <= 100

**Output:**

find the minimum cost required to reach the top of the staircase

Sample test Cases

|  |  |  |
| --- | --- | --- |
|  | Input | Output |
| STC1 | 3  10  15  20 | 15 |
| STC2 | 1  100  1  1  1  100  1  1  100  1 | 6 |

**Solution 16:**

#include <stdio.h>

#include <stdlib.h>

int minCostClimbingStairs(int\* cost, int n) {

int i, oneStep = 0, twoStep = 0;

for(i = n-1; i >= 0; i--) {

int current = cost[i] + ((oneStep < twoStep) ? oneStep : twoStep);

twoStep = oneStep;

oneStep = current;

}

return (oneStep < twoStep) ? oneStep : twoStep;

}

int main() {

int n, i;

scanf("%d", &n);

int \*cost = (int \*) malloc(n \* sizeof(int));

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

scanf("%d", &cost[i]);

}

int minCost = minCostClimbingStairs(cost, n);

printf("%d", minCost);

free(cost);

return 0;

}

Test Cases

|  |  |  |
| --- | --- | --- |
|  | Input | Output |
| TC1 | 5  3 4 5 6 7 | 10 |
| TC2 | 4  10 10 10 10 | 20 |
| TC3 | 6  98 76 56 78 98 76 | 230 |
| TC4 | 6  98 89 99 87 99 99 | 275 |
| TC5 | 4  0 0 0 0 | 0 |

Q17. Define a structure named Time to represent the time of day (hour, minute, and second). Write a program to read two times from the user, add them, and display the result on the console.

**Input:**

two lines with three integers hour, minute, second.

**Constraints:**

0<=**minute,second**<=60

0<=**hours**<=24

**Output:**

**print the added time in the format hours:minutes:seconds**

Sample test Cases

|  |  |  |
| --- | --- | --- |
|  | Input | Output |
| STC1 | 12 23 45  21 43 45 | 34:7:30 |
| STC2 | 20 10 10  2 20 30 | 22:30:40 |

**Solution 17:**

#include <stdio.h>

struct Time {

int hour;

int minute;

int second;

};

int main() {

struct Time t1, t2, result;

scanf("%d%d%d", &t1.hour, &t1.minute, &t1.second);

scanf("%d%d%d", &t2.hour, &t2.minute, &t2.second);

result.second = t1.second + t2.second;

result.minute = t1.minute + t2.minute + result.second/60;

result.hour = t1.hour + t2.hour + result.minute/60;

result.second %= 60;

result.minute %= 60;

printf("%d:%d:%d\n", result.hour, result.minute, result.second);

return 0;

}

Test Cases

|  |  |  |
| --- | --- | --- |
|  | Input | Output |
| TC1 | 12 40 60  21 23 43 | 34:4:43 |
| TC2 | 0 0 0  0 0 0 | 0:0:0 |
| TC3 | 12 23 34  0 0 0 | 12:23:34 |
| TC4 | 23 34 45  12 23 45 | 41:1:20 |
| TC5 | 20 30 40  20 30 40 | 1 |

**SECTION-D (Coding Question)(1x10 mark=10 mark)**

Q18 **Suppose you are working on a project that involves processing large matrices. One of the tasks required is to calculate the sum of non-diagonal elements of a given m x n matrix. Since the size of the matrix can be quite large, you decide to allocate memory dynamically to ensure efficient use of memory. To accomplish this task, you decide to write a 'C' program that takes in the dimensions of the matrix as input and dynamically allocates memory to create the matrix. The program then calculates the sum of all the non-diagonal elements and outputs the result.**

**Sample Input**:

Enter Limit for Rows : 3 Enter Limit for Columns : 3

Enter Elements for Matrix of Size 3\*3:

1 2 3 4 5 6 7 8 9

**Sample Output**:

Non-Diagonal Elements are: 2 3 4 6 7 8 Sum of Non-Diagonal Elements in Matrix = 30

|  |  |  |  |
| --- | --- | --- | --- |
|  | Test Case 1 | Test Case 2 | Test Case 3 |
| Input | Enter Limit for Rows : 4 Enter Limit for Columns : 4  Enter Elements for Matrix of Size 4\*4:  1 3 5 7 2 4 6 8 1 2 3 4 9 8 7 6 | Enter Limit for Rows : 2 Enter Limit for Columns : 2  Enter Elements for Matrix of Size 2\*2:  2 4 6 8 | Enter Limit for Rows : 3 Enter Limit for Columns : 3  Enter Elements for Matrix of Size 3\*3:  1 2 3 4 5 6 7 8 9 |
| Output | Non-Diagonal Elements are: 3 5 7 2 6 8 1 3 4 9 7 6 Sum of Non-Diagonal Elements in Matrix = 51 | Non-Diagonal Elements are: 4 6 Sum of Non-Diagonal Elements in Matrix = 10 | Non-Diagonal Elements are: 2 3 4 6 7 8 Sum of Non-Diagonal Elements in Matrix = 30 |

**Solution:**

#include<stdio.h>

#include<stdlib.h>

int main()

{

int \*\*a, row,col,i,j,s=0;

printf("Enter Limit for Rows : ");

scanf("%d",&row);

printf("\nEnter Limit for Columns : ");

scanf("%d",&col);

a=(int \*\*)malloc(row\*sizeof(int\*));

for(i=0;i<row;i++)

{

a[i]=(int \*)malloc(col\*sizeof(int));

}

printf("\nEnter Elements for Matrix of Size %d\*%d:\n\n",row,col);

for(i=0;i<row;i++)

{

for(j=0;j<col;j++)

{

scanf("%d",&a[i][j]);

}

}

printf("\n%d\*%d Matrix : \n\n",row,col);

for(i=0;i<row;i++)

{

for(j=0;j<col;j++)

{

printf("%3d ",a[i][j]);

}

printf("\n");

}

printf("\nNon-Diagonal Elements are: ");

for(i=0;i<row;i++)

{

for(j=0;j<col;j++)

{

if(i!=j)

{

printf("%d ",a[i][j]); //Minor diagonal not considered here

s=s+a[i][j];

}

}

}

printf("\n\nSum of Non-Diagonal Elements in Matrix = %d",s);

return 0;

}