**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 Which of the following is true about recursion?

a) It is faster than iteration

b) It is slower than iteration

c) It takes more memory than iteration

d) It takes less memory than iteration

Answer: c) It takes more memory than iteration

Q2 What is tail recursion?

a) When the recursive call is the last statement in the function

b) When the recursive call is the first statement in the function

c) When the recursive call is in the middle of the function

d) None of the above

Answer: a) When the recursive call is the last statement in the function

Q3 What is the maximum number of recursive calls that can be made in a program?

It depends on the size of the stack

It depends on the size of the heap

It is unlimited

None of the above

Answer: a) It depends on the size of the stack

Q4 1. What is the maximum number of bits that can be used in a bit field in C?

A) 8

B) 16

C) 32

D) There is no maximum limit

Answer: D) There is no maximum limit

Q51. Which of the following is an example of a bit field in a struct in C?

A) struct { int x: 4; int y: 4; };

B) struct { char a; char b; };

C) struct { int x; int y; };

D) None of the above

Answer: A) struct { int x: 4; int y: 4; };

Q6 1. Which of the following is a valid way to declare a pointer to a struct in C?

A) struct \*p;

B) struct p;

C) struct \*\*p;

D) struct mystruct \*p;

Answer: D) struct mystruct \*p;

Q7 1. Which of the following statements is true about enumerations in C?

a. Enumerations can be used to define integer constants with meaningful names

b. Enumerations can be used to define floating-point constants with meaningful names

c. Enumerations can be used to define string constants with meaningful names

d. Enumerations can be used to define boolean constants with meaningful names

Answer: a

Q8 1. Which of the following keywords is used to define a constant in an enumeration in C?

a. enum

b. typedef

c. const

d. None of the above

Answer: d

Q9 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);

Q10 1. What is the purpose of the remove() function in C file handling?

a) to delete a file

b) to rename a file

c) to create a new file

d) to open a file

Answer: a) to delete a file

**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 3?

int mystery(int n) {

if (n <= 0) {

return 0;

}

return mystery(n-1) + n;

}

A. 1

B. 2

C. 3

D. 6

Q12 What is the output of the following code?

#include <stdio.h>

struct point {

int x;

int y;

};

int main() {

struct point p1 = { 5, 10 };

struct point p2 = { 15, 20 };

if (p1.x == p2.x && p1.y == p2.y) {

printf("Points are equal\n");

}

else {

printf("Points are not equal\n");

}

return 0;

}

A. Points are equal

B. Points are not equal

C. Error

D. None of the above

Correct answer: B

Q13 What does the sizeof operator return when used with a typedef structure?

a. The size of the structure type

b. The size of a variable of the structure type

c. The size of a pointer to the structure type

d. The size of the first member of the structure type

Correct answer: b

Q14 What is the output of the following code snippet?

#include <stdio.h>

enum { A = -1, B, C = 2 };

int main() {

printf("%d %d %d", A, B, C);

return 0;

}

a) -1 0 1

b) -1 0 2

c) -1 1 2

d) Compiler error

Answer: b

Q15 What is the output of the following program?

#include <stdio.h>

struct employee {

char name[50];

int age;

float salary;

};

int main() {

struct employee employees[3] = {{"John Doe", 25, 5000.0}, {"Jane Smith", 30, 7000.0}, {"Bob Johnson", 35, 9000.0}};

FILE \*fp = fopen("employees.txt", "w");

fwrite(employees, sizeof(struct employee), 3, fp);

fclose(fp);

fp = fopen("employees.txt", "r");

struct employee e[3];

fread(e, sizeof(struct employee), 3, fp);

fclose(fp);

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

printf("Employee %d: %s %d %.2f\n", i+1, e[i].name, e[i].age, e[i].salary);

}

return 0;

}

a) The program writes the information of 3 employees to a file named employees.txt and then reads it back and prints the information on the screen.

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. Consider a game where a player can score 2 or 3 or 5 points in a move. Given a total score n, find a number of distinct combinations to reach the given score.

**Input:**

One line with integer **n**

**Constraints:**

1<=**n**<=100

**Output:**

**a number of distinct combinations to reach the given score**

Sample test Cases

|  |  |  |
| --- | --- | --- |
|  | Input | Output |
| STC1 | 5 | 3 |
| STC2 | 3 | 1 |

**Solution 16:**

#include <stdio.h>

int countCombinations(int n) {

if (n == 0) {

return 1;

} else if (n < 0) {

return 0;

} else {

return countCombinations(n - 2) + countCombinations(n - 3) + countCombinations(n - 5);

}

}

int main() {

int n ;

scanf("%d",&n);

int numCombinations = countCombinations(n);

printf("%d",numCombinations);

return 0;

}

Test Cases

|  |  |  |
| --- | --- | --- |
|  | Input | Output |
| TC1 | 3 | 1 |
| TC2 | 45 | 3339800 |
| TC3 | 34 | 65766 |
| TC4 | 0 | 1 |
| TC5 | 1 | 0 |

Q17. Create a structure to represent a rectangle with fields for width and height. Write a program that prompts the user to enter the dimensions of two rectangles, calculates the area of each rectangle, and then prints the dimensions and areas of both rectangles.

**Input:**

**two lines containing two integers width and height.**

**Constraints:**

0<=**width,height**<=1000

**Output:**

**two lines giving area of rectangle1 and rectangle2 respectively.**

Sample test Cases

|  |  |  |
| --- | --- | --- |
|  | Input | Output |
| STC1 | 3 5  5 6 | 15  30 |
| STC2 | 1 0  8 7 | 0  56 |

**Solution 16:**

#include <stdio.h>

struct rectangle {

int width;

int height;

};

int main() {

struct rectangle rect1, rect2;

int area1, area2;

scanf("%d%d", &rect1.width, &rect1.height);

scanf("%d%d", &rect2.width, &rect2.height);

area1 = rect1.width \* rect1.height;

area2 = rect2.width \* rect2.height;

printf("%d\n",area1);

printf("%d",area2);

return 0;

}

Test Cases

|  |  |  |
| --- | --- | --- |
|  | Input | Output |
| TC1 | 123 23  324 334 | 2829  108216 |
| TC2 | 21 43  23 45 | 903  1035 |
| TC3 | 324 444  343 454 | 143856  155722 |
| TC4 | 999 999  999 978 | 998001  977022 |
| TC5 | 1 1  1 1 | 1  1 |

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

Q18 **Suppose you are implementing a program to print a stack of books in reverse order, where each book is represented as a node in a linked list. You want to use a stack data structure to keep track of the books as you traverse the linked list. However, the stack is currently in the wrong order, so you need to reverse it before printing. To accomplish this, you decide to write a C program that uses recursion to reverse the stack.**

**Sample Input**:

Enter length of list: 5

**Sample Output**:

The sequence of contents in stack : 5 4 3 2 1

Inversing the contents of the stack

The contents in stack after reversal 1 2 3 4 5

|  |  |  |  |
| --- | --- | --- | --- |
|  | Test Case 1 | Test Case 2 | Test Case 3 |
| Input | Enter length of list: 0 | Enter length of list: 1 | Enter length of list: 5 |
| Output | The sequence of contents in stack :  Inversing the contents of the stack  The contents in stack after reversal | The sequence of contents in stack : 1  Inversing the contents of the stack  The contents in stack after reversal 1 | The sequence of contents in stack : 5 4 3 2 1  Inversing the contents of the stack  The contents in stack after reversal 1 2 3 4 5 |

**Solution:**

#include <stdio.h>

#include <stdlib.h>

struct node

{

int a;

struct node \*next;

};

void generate(struct node \*\*);

void display(struct node \*);

void stack\_reverse(struct node \*\*, struct node \*\*);

void delete(struct node \*\*);

int main()

{

struct node \*head = NULL;

generate(&head);

printf("\nThe sequence of contents in stack : \n");

display(head);

printf("\n\nInversing the contents of the stack\n");

if (head != NULL)

{

stack\_reverse(&head, &(head->next));

}

printf("\nThe contents in stack after reversal\n");

display(head);

delete(&head);

return 0;

}

void stack\_reverse(struct node \*\*head, struct node \*\*head\_next)

{

struct node \*temp;

if (\*head\_next != NULL)

{

temp = (\*head\_next)->next;

(\*head\_next)->next = (\*head);

\*head = \*head\_next;

\*head\_next = temp;

stack\_reverse(head, head\_next);

}

}

void display(struct node \*head)

{

if (head != NULL)

{

printf("%d ", head->a);

display(head->next);

}

}

void generate(struct node \*\*head)

{

int num, i;

struct node \*temp;

printf("Enter length of list: ");

scanf("%d", &num);

for (i = num; i > 0; i--)

{

temp = (struct node \*)malloc(sizeof(struct node));

temp->a = i;

if (\*head == NULL)

{

\*head = temp;

(\*head)->next = NULL;

}

else

{

temp->next = \*head;

\*head = temp;

}

}

}

void delete(struct node \*\*head)

{

struct node \*temp;

while (\*head != NULL)

{

temp = \*head;

\*head = (\*head)->next;

free(temp);

}

}