**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. Which of the following is true about the stack in recursion?

a) It is used to store the return addresses

b) It is used to store the local variables

c) It is used to store the function arguments

d) All of the above

Answer: d) All of the above

Q2 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)

Q3 1. Which operator is used to access members of a struct in C?

A) .

B) ->

C) &

D) \*

Answer: A) .

Q4 1. What is the maximum number of members a struct can have in C?

A) 64

B) 128

C) 256

D) There is no maximum limit

Answer: D) There is no maximum limit

Q5 1. What is the keyword used to allocate memory dynamically for a struct in C?

A) malloc

B) calloc

C) realloc

D) free

Answer: A) malloc

Q6 1. What is the keyword used to define an enumeration in C?

A) enum

B) define

C) typedef

D) struct

Answer: A) enum

Q7 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

Q8 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;

Q9 1. Which keyword is used to declare a union in C?

a. union

b. struct

c. typedef

d. None of the above

Answer: a

Q10 1. Can two elements of an enumeration have the same value in C?

a. Yes

b. No

c. Case specific

d. Enum is not a datatype

Answer: a

**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 qux(int n) {

if (n > 0) {

qux(n-1);

printf("%d ", n);

qux(n-2);

}

return n;

}

A. 1 2 3 1

B. 3 2 1 1

C. 1 3 2 3

D. 2 1 3 2

Q12 What is the output of the following code?

#include <stdio.h>

struct person {

char \*name;

int age;

};

int main() {

struct person p1 = { "John", 25 };

struct person p2 = p1;

p2.name = "Jane";

printf("%s\n", p1.name);

return 0;

}

A. John

B. Jane

C. Error

D. None of the above

Correct answer: A

Q13 Which of the following statements is true about the typedef keyword?

a. It can only be used with structures

b. It creates a new name for an existing type

c. It modifies the structure members

d. It declares a new variable

Correct answer: b

Q14 What will be the output of the following C code?

#include <stdio.h>

enum weekdays { MON, TUE, WED, THU, FRI, SAT, SUN };

int main() {

enum weekdays w1 = TUE, w2 = SUN;

printf("%d %d", w1, w2);

return 0;

}

a) 1 5

**b) 1 6**

c) 1 4

d) 1 3

Q15 What is the output of the following program?

#include <stdio.h>

union test {

int x;

char c;

};

int main() {

union test t;

t.x = 65;

printf("%c\n", t.c);

return 0;

}

a) A

b) 65

c) The program produces an error during compilation.

d) The program produces a runtime error.

Correct answer: a

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

Q16. Define an enum named "Months" with January, February, March, April, May, June, July, August, September, October, November, December as its members. Write a program that takes an integer value from the user and prints the corresponding month.

**Input:**

one integer n.

**Constraints:**

**0 < n<=12**

**Output:**

prints a string (corresponding month).

Sample test Cases

|  |  |  |
| --- | --- | --- |
|  | Input | Output |
| STC1 | 2 | February |
| STC2 | 12 | December |

**Solution 16:**

#include <stdio.h>

enum Months {

January = 1, February, March, April, May, June, July, August, September, October, November, December

};

int main() {

int monthNum;

printf("Enter a month number (1-12): ");

scanf("%d", &monthNum);

switch (monthNum) {

case January:

printf("January");

break;

case February:

printf("February");

break;

case March:

printf("March");

break;

case April:

printf("April");

break;

case May:

printf("May");

break;

case June:

printf("June");

break;

case July:

printf("July");

break;

case August:

printf("August");

break;

case September:

printf("September");

break;

case October:

printf("October");

break;

case November:

printf("November");

break;

case December:

printf("December");

break;

default:

printf("Invalid month number");

break;

}

return 0;

}

Test Cases

|  |  |  |
| --- | --- | --- |
|  | Input | Output |
| TC1 | 5 | May |
| TC2 | 7 | July |
| TC3 | 8 | August |
| TC4 | 9 | September |
| TC5 | 1 | January |

Q17. The atoi() function takes a string (which represents an integer) as an argument and returns its value. How to compute recursively.

**Input:**

one string

**Constraints:**

0 < length < 7

**Output:**

**an Integer**

Sample test Cases

|  |  |  |
| --- | --- | --- |
|  | Input | Output |
| STC1 | 123 | 123 |
| STC2 | 011 | 11 |

**Solution 17:**

**#include <stdio.h>**

**#include <string.h>**

**int myAtoiRecursive(char\* str, int n)**

**{**

**int count = 0, check;**

**for (int i = 0; i <= strlen(str); ++i)**

**{**

**check = ( ( 'a' <= str[i] && str[i] <= 'z' ) ||**

**( 'A' <= str[i] && str[i] <= 'Z' ) )?1:0;**

**if (check)**

**{**

**++count;**

**}**

**}**

**if (count != 0) {**

**return 0;**

**}**

**if (n == 1)**

**return \*str - '0';**

**return (10 \* myAtoiRecursive(str, n - 1) + str[n - 1]**

**- '0');**

**}**

**// Driver Program**

**int main(void)**

**{**

**char str[10];**

**gets(str);**

**int n = strlen(str);**

**printf("%d", myAtoiRecursive(str, n));**

**return 0;**

**}**

Test Cases

|  |  |  |
| --- | --- | --- |
|  | Input | Output |
| TC1 | 1001 | 1001 |
| TC2 | 9999999 | 9999999 |
| TC3 | 0 | 0 |
| TC4 | 7687 | 7687 |
| TC5 | 132434 | 132434 |

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

Q18 **Imagine you are building a calculator application where the user can enter a number and perform various operations on it. To allow the user to perform calculations on the reverse of the entered number, you decide to write a function to find the reverse of the number using recursion. This function will take the user input as an argument, find its reverse using recursion, and return the reversed number for further calculations.**

**Sample Input**:

Enter an integer number to reverse: 12345

**Sample Output**:

The reverse of 12345 is 54321

|  |  |  |  |
| --- | --- | --- | --- |
|  | Test Case 1 | Test Case 2 | Test Case 3 |
| Input | Enter an integer number to reverse: 12345 | Enter an integer number to reverse: 987654321 | nter an integer number to reverse: 100 |
| Output | The reverse of 12345 is 54321. | The reverse of 987654321 is 123456789. | The reverse of 100 is 001 |

**Solution:**

#include <stdio.h>

#include <math.h>

int rev(int, int);

int main()

{

int num, result;

int length = 0, temp;

printf("Enter an integer number to reverse: ");

scanf("%d", &num);

temp = num;

while (temp != 0)

{

length++;

temp = temp / 10;

}

result = rev(num, length);

printf("The reverse of %d is %d.\n", num, result);

return 0;

}

int rev(int num, int len)

{

if (len == 1)

{

return num;

}

else

{

return (((num % 10) \* pow(10, len - 1)) + rev(num / 10, --len));

}

}