1. Recursion is a method in which the solution of a problem depends on \_\_\_\_\_\_\_\_\_\_\_\_  
   a) Larger instances of different problems  
   b) Larger instances of the same problem  
   c) Smaller instances of the same problem  
   d) Smaller instances of different problems

Answer: smaller instance of same problem.

1. Which of the following problems can’t be solved using recursion?  
   a) Factorial of a number  
   b) Nth fibonacci number  
   c) Length of a string  
   d) Problems without base case

Answer: problems without a base case.

1. Recursion is similar to which of the following?  
   a) Switch Case  
   b) Loop  
   c) If-else  
   d) if elif else

It is similar to loop case.

1. In recursion, the condition for which the function will stop calling itself is \_\_\_\_\_\_\_\_\_\_\_\_  
   a) Best case  
   b) Worst case  
   c) Base case  
   d) There is no such condition

Answer : base case.

\*5.What will happen when the below code snippet is executed?

void my\_recursive\_function()

{

my\_recursive\_function();

}

int main()

{

my\_recursive\_function();

return 0;

}

Answer: the code will run for some time and will stop when stack overflows.

1. What is the output of the following code?
2. void my\_recursive\_function(int n)
3. {
4. if(n == 0)
5. return;
6. printf("%d ",n);
7. my\_recursive\_function(n-1);
8. }
9. int main()
10. {
11. my\_recursive\_function(10);
12. return 0;
13. }

Answer : 10,9,8……1

7.What is the base case for the following code?

void my\_recursive\_function(int n)

{

if(n == 0)

return;

printf("%d ",n);

my\_recursive\_function(n-1);

}

int main()

{

my\_recursive\_function(10);

return 0;

}

8. How many times is the recursive function called, when the following code is executed?

void my\_recursive\_function(int n)

{

if(n == 0)

return;

printf("%d ",n);

my\_recursive\_function(n-1);

}

int main()

{

my\_recursive\_function(10);

return 0;

}

Answer: 11

9. What does the following recursive code do?

void my\_recursive\_function(int n)

{

if(n == 0)

return;

my\_recursive\_function(n-1);

printf("%d ",n);

}

int main()

{

my\_recursive\_function(10);

return 0;

}

Answer: print numbers from 1 to 10.

10. Which of the following statements is true?  
a) Recursion is always better than iteration  
b) Recursion uses more memory compared to iteration  
c) Recursion uses less memory compared to iteration  
d) Iteration is always better and simpler than recursion

Ansewer: b

Because recursion uses stack memory.

11. What will be the output of the following code?

int cnt=0;

void my\_recursive\_function(int n)

{

if(n == 0)

return;

cnt++;

my\_recursive\_function(n/10);

}

int main()

{

my\_recursive\_function(123456789);

printf("%d",cnt);

return 0;

}

Answer : 10

12. What will be the output of the following code?

void my\_recursive\_function(int n)

{

if(n == 0)

{

printf("False");

return;

}

if(n == 1)

{

printf("True");

return;

}

if(n%2==0)

my\_recursive\_function(n/2);

else

{

printf("False");

return;

}

}

int main()

{

my\_recursive\_function(100);

return 0;

}

Answer : false

13. What is the output of the following code?

int cnt = 0;

void my\_recursive\_function(char \*s, int i)

{

if(s[i] == '**\0**')

return;

if(s[i] == 'a' || s[i] == 'e' || s[i] == 'i' || s[i] == 'o' || s[i] == 'u')

cnt++;

my\_recursive\_function(s,i+1);

}

int main()

{

my\_recursive\_function("thisisrecursion",0);

printf("%d",cnt);

return 0;

}

1. 6  
   b) 9  
   c) 5  
   d) 10

Answer: 6 due to 6 vowels.

14. What is the output of the following code?

void my\_recursive\_function(int \*arr, int val, int idx, int len)

{

if(idx == len)

{

printf("-1");

return ;

}

if(arr[idx] == val)

{

printf("%d",idx);

return;

}

my\_recursive\_function(arr,val,idx+1,len);

}

int main()

{

int array[10] = {7, 6, 4, 3, 2, 1, 9, 5, 0, 8};

int value = 2;

int len = 10;

my\_recursive\_function(array, value, 0, len);

return 0;

}

a) 3  
b) 4  
c) 5  
d) 6

answer: b