I

Tail Recursion

What is tail recursion?

A recursive function is tail recursive when recursive call is the last thing executed by the function. For example the following C++ function print() is tail recursive.

```
// An example of tail recursive function
void print(int n)
{
      if (n < 0)  return;
      cout << " " << n;

      // The last executed statement is recursive call
      print(n-1);
}
```

Why do we care?

The tail recursive functions considered better than non tail recursive functions as tail-recursion can be optimized by compiler. The idea used by compilers to optimize tail-recursive functions is simple, since the recursive call is the last statement, there is nothing left to do in the current function, so saving the current function's stack frame is of no use (See this for more details).

Can a non-tail recursive function be written as tail-recursive to optimize it?

Consider the following function to calculate factorial of n. It is a non-tail-recursive function. Although it looks like a tail recursive at first look. If we take a closer look, we can see that the value returned by fact(n-1) is used in fact(n), so the call to fact(n-1) is not the last thing done by fact(n)

- C++

- Java
- Python 3
- C#
- PHP

```
#include<iostream>
using namespace std;

// A NON-tail-recursive function.  The function is not tail
// recursive because the value returned by fact(n-1) is used in
// fact(n) and call to fact(n-1) is not the last thing done by
fact(n)
unsigned int fact(unsigned int n)
```

```
{
    if (n == 0) return 1;

    return n*fact(n-1);
}

// Driver program to test above function
int main()
{
    cout << fact(5);
    return 0;
}
```

Output :

```
120
```

The above function can be written as a tail recursive function. The idea is to use one more argument and accumulate the factorial value in second argument. When n reaches 0, return the accumulated value.

- C++
- Java
- Python 3
- C#
- PHP

```
#include<iostream>
using namespace std;

// A tail recursive function to calculate factorial
unsigned factTR(unsigned int n, unsigned int
a)
{
    if (n == 0)  return a;

    return factTR(n-1, n*a);
}

// A wrapper over factTR
unsigned int fact(unsigned int n)
{
   return factTR(n, 1);
}

// Driver program to test above function
int main()
{
```

```
    cout << fact(5);
    return 0;
}
```

Tail Call Elimination

We have discussed (in tail recursion) that a recursive function is tail recursive if recursive call is the last thing executed by the function.

```
// An example of tail recursive function
void print(int n)
{
    if (n < 0)
        return;
    cout << " " << n;

    // The last executed statement is recursive call
    print(n-1);
}
```

We also discussed that a tail recursive is better than non-tail recursive as tail-recursion can be optimized by modern compilers. Modern compiler basically do tail call elimination to optimize the tail recursive code.

If we take a closer look at above function, we can remove the last call with goto. Below are examples of tail call elimination.

```
// Above code after tail call elimination
void print(int n)
{
start:
    if (n < 0)
        return;
    cout << " " << n;

    // Update parameters of recursive call
    // and replace recursive call with goto
    n = n-1
    goto start;
}
```

In this example the user input is broken into its corresponding percentage value for 1%age to 100% using recursion.

```
#include<stdio.h>

//function declaration
int RecursvFcnTogetPrcntge();

float Var;
unsigned int count=1;
float Prcntge;
```

```c
int main()
{

    printf("\nEnter a value to split in percentage: ");
    scanf("%f",&Var);

    Var=Var/100;
    RecursvFcnTogetPrcntge();
    printf("\n");

    return 0;
}

int RecursvFcnTogetPrcntge()
{

    if(count==101)
    {
        return 1;
    }

    Prcntge=Var*count;
    printf("\n%3d Percent = %.02f",count, Prcntge);
    count++;
    RecursvFcnTogetPrcntge();
    return 0;
}
```

Calculate power of a number program using recursion.

```c
 /*C program to calculate power of any number using recursion*/
1
2#include <stdio.h>
3
4//function for calculating power
5long int getPower(int b,int p)
6{
7      long int result=1;
8      if(p==0) return result;
9      result=b*(getPower(b,p-1));  //call function again
10}
11int main()
12{
13      int base,power;
14      long int result;
15
16      printf("Enter value of base: ");
17      scanf("%d",&base);
18
19      printf("Enter value of power: ");
20      scanf("%d",&power);
21
22      result=getPower(base,power);
23
24      printf("%d to the power of %d is:
25%ld\n",base,power,result);
26
27      return 0;
28  }
```

/*C program to print fibonacii series till N terms.*/

```c
#include <stdio.h>

//function to print fibonacii series
void getFibonacii(int a,int b, int n)
{
    int sum;
    if(n>0)
    {
        sum=a+b;
        printf("%d ",sum);
        a=b;
        b=sum;
        getFibonacii(a,b,n-1);
    }
```

```
}
int main()
{
    int a,b,sum,n;
    int i;

    a=0;        //first term
    b=1;        //second term

    printf("Enter total number of terms: ");
    scanf("%d",&n);

    printf("Fibonacii series is : ");
    //print a and b as first and second terms of series
    printf("%d\t%d\t",a,b);
    //call function with (n-2) terms
    getFibonacii(a,b,n-2);
    printf("\n");
    return 0;
}
```

Program to count digits in C using recursion

```
/*C program to count digits using recursion.*/

#include <stdio.h>

//function to count digits
int countDigits(int num)
{
    static int count=0;

    if(num>0)
    {
        count++;
        countDigits(num/10);
    }
    else
    {
        return count;
    }
}
```

```c
int main()
{
    int number;
    int count=0;

    printf("Enter a positive integer number: ");
    scanf("%d",&number);

    count=countDigits(number);

    printf("Total digits in number %d is: %d\n",number,count);

    return 0;
}
```

Sum of digits of a number program using recursion.

```c
/*C program to find sum of all digits using recursion.*/

#include <stdio.h>

//function to calculate sum of all digits
int sumDigits(int num)
{
    static int sum=0;
    if(num>0)
    {
        sum+=(num%10); //add digit into sum
        sumDigits(num/10);
    }
    else
    {
        return sum;
    }
}
int main()
{
    int number,sum;

    printf("Enter a positive integer number: ");
    scanf("%d",&number);
```

```
    sum=sumDigits(number);

    printf("Sum of all digits are: %d\n",sum);

    return 0;
}
```

Length of the string program using recursion

```
/*function to calculate length of the string using recursion.*/

#include <stdio.h>

//function to calculate length of the string using recursion
int stringLength(char *str)
{
    static int length=0;
    if(*str!=NULL)
    {
        length++;
        stringLength(++str);
    }
    else
    {
        return length;
    }
}
int main()
{
    char str[100];
    int length=0;

    printf("Enter a string: ");
    gets(str);

    length=stringLength(str);

    printf("Total number of characters (string length) are:
%d\n",length);

    return 0;
}
```

C Program to Reverse a Stack using Recursion

This C program, using recursion, reverses a stack content. Stack here is represented using a linked list. A linked list is an ordered set of data elements, each containing a link to its successor.
Here is the source code of the C program to display a linked list in reverse. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2.  * C Program to Reverse a Stack using Recursion
3.  */
4. #include <stdio.h>
5. #include <stdlib.h>
6.
7. struct node
8. {
9.     int a;
10.     struct node *next;
11.};
12.
13.void generate(struct node **);
14.void display(struct node *);
15.void stack_reverse(struct node **, struct node **);
16.void delete(struct node **);
17.
18.int main()
19.{
20.     struct node *head = NULL;
21.
22.     generate(&head);
23.     printf("\nThe sequence of contents in stack\n");
24.     display(head);
25.     printf("\nInversing the contents of the stack\n");
26.     if (head != NULL)
27.     {
28.         stack_reverse(&head, &(head->next));
29.     }
30.     printf("\nThe contents in stack after reversal\n");
31.     display(head);
32.     delete(&head);
33.
34.     return 0;
35.}
36.
37.void stack_reverse(struct node **head, struct node **head_next)
38.{
39.     struct node *temp;
40.
41.     if (*head_next != NULL)
42.     {
```

```
43.             temp = (*head_next)->next;
44.             (*head_next)->next = (*head);
45.             *head = *head_next;
46.             *head_next = temp;
47.             stack_reverse(head, head_next);
48.         }
49.}
50.
51.void display(struct node *head)
52.{
53.         if (head != NULL)
54.         {
55.             printf("%d  ", head->a);
56.             display(head->next);
57.         }
58.}
59.
60.void generate(struct node **head)
61.{
62.         int num, i;
63.         struct node *temp;
64.
65.         printf("Enter length of list: ");
66.         scanf("%d", &num);
67.         for (i = num; i > 0; i--)
68.         {
69.             temp = (struct node *)malloc(sizeof(struct node));
70.             temp->a = i;
71.             if (*head == NULL)
72.             {
73.                 *head = temp;
74.                 (*head)->next = NULL;
75.             }
76.             else
77.             {
78.                 temp->next = *head;
79.                 *head = temp;
80.             }
81.         }
82.}
83.
84.void delete(struct node **head)
85.{
86.         struct node *temp;
87.         while (*head != NULL)
88.         {
89.             temp = *head;
90.             *head = (*head)->next;
```

```
91.             free(temp);
92.         }
93.}
```

C Program to Find the Length of the String using Recursion

This C Program uses recursive function & counts the number of nodes in a linked list. A linked list is an ordered set of data elements, each containing a link to its successor.

Here is the source code of the C program to count the number of nodes in a linked list. The C Program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2.  * Recursive C program to find length of a linked list
3.  */
4. #include <stdio.h>
5. #include <stdlib.h>
6.
7. struct node
8. {
9.     int a;
10.     struct node *next;
11.};
12.
13.void generate(struct node **);
14.int length(struct node*);
15.void delete(struct node **);
16.
17.int main()
18.{
19.     struct node *head = NULL;
20.     int count;
21.
22.     generate(&head);
23.     count = length(head);
24.     printf("The number of nodes are: %d\n", count);
25.     delete(&head);
26.     return 0;
27.}
28.
29.void generate(struct node **head)
30.{
31.     /* for unknown number of nodes use num = rand() % 20; */
32.     int num = 10, i;
33.     struct node *temp;
34.
35.     for (i = 0; i < num; i++)
36.     {
37.         temp = (struct node *)malloc(sizeof(struct node));
```

```
38.             temp->a = i;
39.             if (*head == NULL)
40.             {
41.                     *head = temp;
42.                     (*head)->next = NULL;
43.             }
44.             else
45.             {
46.                     temp->next = *head;
47.                     *head = temp;
48.             }
49.     }
50.}
51.
52.int length(struct node *head)
53.{
54.     if (head->next == NULL)
55.     {
56.             return 1;
57.     }
58.     return (1 + length(head->next));
59.}
60.
61.void delete(struct node **head)
62.{
63.     struct node *temp;
64.     while (*head != NULL)
65.     {
66.             temp = *head;
67.             *head = (*head)->next;
68.             free(temp);
69.     }
70.}
```

C Program to Find the Biggest Number in an Array of Numbers using Recursion

This is a C Program which prints the largest number in an unsorted array of elements using recursion.

Problem Description

This program will implement a one-dimentional array defining elements in unsorted fashion, in which we need to find largest element using Recursion method. The array used here is of type integer.

Problem Solution

1. Create an array, taking its size from the users and define all its elements.
2. Now make a function passing three parameters, array, last index of array and largest element of the array.
3. Assuming first element to be the largest element in the array, call this function.
4. Inside this function, since first element has been selected as largest for now, check if the element at index passed to this function (initially last index) is greater than the largest number passed. Update the largest number and proceed to call the same function, but now with index-1.
5. This way each element is checked from index (size-1) to 1 is compared with largest number. The base condition for this recursive function would be to return the largest number when it reaches the first index.

Program/Source Code

Here is the source code of the C program to print the largest number in an unsorted array. The program is successfully compiled and tested using Turbo C compiler in windows environment. The program output is also shown below.

```
1.
2.      /*
3.       * C Program to find the Biggest Number in an Array of Numbers using
4.       * Recursion
5.       */
6.
7.      #include <stdio.h>
8.      int large(int[], int, int);
9.
10.      int main()
11.      {
12.
13.              int size;
14.              int largest;
15.              int list[20];
16.              int i;
17.
18.              printf("Enter size of the list:");
19.              scanf("%d", &size);
20.
21.              printf("Printing the list:\n");
22.              for (i = 0; i < size ; i++)
23.              {
```

```
24.                 list[i] = rand() % size;
25.                 printf("%d \t", list[i]);
26.             }
27.
28.             if (size == 0)
29.             {
30.                 printf("Empty list\n");
31.             }
32.
33.             else
34.             {
35.                 largest = list[0];
36.                 largest = large(list, size - 1, largest);
37.                 printf("\nThe largest number in the list is: %d\n", largest);
38.             }
39.
40.     }
41.
42.     int large(int list[], int position, int largest)
43.     {
44.
45.             if (position == 0)
46.                 return largest;
47.
48.             if (position > 0)
49.             {
50.                 if (list[position] > largest)
51.                 {
52.                     largest = list[position];
53.                 }
54.                 return large(list, position - 1, largest);
55.             }
56.
57.     }
```

C Program to Display the Nodes of a Linked List in Reverse using Recursion

This C Program uses recursive function & reverses the nodes in a linked list and displays the list. A linked list is an ordered set of data elements, each containing a link to its successor. This program makes each data element to link to its predecessor.

Here is the source code of the C program to reverse the nodes and display the linked list. The C Program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2.  * Recursive C program to reverse nodes of a linked list and display
3.  * them
4.  */
5. #include <stdio.h>
```

```c
#include <stdlib.h>

struct node
{
	int data;
	struct node *next;
};

void print_reverse_recursive (struct node *);
void print (struct node *);
void create_new_node (struct node *, int );

//Driver Function
int main ()
{
	struct node *head = NULL;
	insert_new_node (&head, 1);
	insert_new_node (&head, 2);
	insert_new_node (&head, 3);
	insert_new_node (&head, 4);
	printf ("LinkedList : ");
	print (head);
	printf ("\nLinkedList in reverse order : ");
	print_reverse_recursive (head);
	printf ("\n");
	return 0;
}

//Recursive Reverse
void print_reverse_recursive (struct node *head)
{
	if (head == NULL)
	{
		return;
	}

	//Recursive call first
	print_reverse_recursive (head -> next);
	//Print later
	printf ("%d ", head -> data);
}

//Print the linkedlist normal
void print (struct node *head)
{
	if (head == NULL)
	{
		return;
```

```
54.       }
55.       printf ("%d ", head -> data);
56.       print (head -> next);
57.}
58.
59.//New data added in the start
60.void insert_new_node (struct node ** head_ref, int new_data)
61.{
62.       struct node * new_node = (struct node *) malloc (sizeof (struct node));
63.       new_node -> data = new_data;
64.       new_node -> next = (*head_ref);
65.       (*head_ref) = new_node;
66.}
```

C Program to Display all the Nodes in a Linked List using Recursion

This C Program uses recursive function & displays a linked list. A linked list is an ordered set of data elements, each containing a link to its successor.
Here is the source code of the C program to display a linked list. The C Program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2.  * Recursive C program to display members of a linked list
3.  */
4. #include <stdio.h>
5. #include <stdlib.h>
6.
7. struct node
8. {
9.       int a;
10.       struct node *next;
11.};
12.
13.void generate(struct node **);
14.void display(struct node*);
15.void delete(struct node **);
16.
17.int main()
18.{
19.       struct node *head = NULL;
20.
21.       generate(&head);
22.       display(head);
23.       delete(&head);
24.       return 0;
```

```
25.}
26.
27.void generate(struct node **head)
28.{
29.	int num = 10, i;
30.	struct node *temp;
31.
32.	for (i = 0; i < num; i++)
33.	{
34.		temp = (struct node *)malloc(sizeof(struct node));
35.		temp->a = i;
36.		if (*head == NULL)
37.		{
38.			*head = temp;
39.			(*head)->next = NULL;
40.		}
41.		else
42.		{
43.			temp->next = *head;
44.			*head = temp;
45.		}
46.	}
47.}
48.
49.void display(struct node *head)
50.{
51.	printf("%d  ", head->a);
52.	if (head->next == NULL)
53.	{
54.		return;
55.	}
56.	display(head->next);
57.}
58.
59.void delete(struct node **head)
60.{
61.	struct node *temp;
62.	while (*head != NULL)
63.	{
64.		temp = *head;
65.		*head = (*head)->next;
66.		free(temp);
67.	}
68.}
```

C Program Count the Number of Occurrences of an Element in the Linked List using Recursion

This C Program uses recursive function & finds the occurrence for an element in an unsorted list. The user enters the element need to be counted.
Here is the source code of the C program to find the number of occurrences of a given number in a list. The C Program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2.  * C program to find the number of occurences of a given number in a
3.  * list
4.  */
5. #include <stdio.h>
6.
7. void occur(int [], int, int, int, int *);
8.
9. int main()
10.{
11.     int size, key, count = 0;
12.     int list[20];
13.     int i;
14.
15.     printf("Enter the size of the list: ");
16.     scanf("%d", &size);
17.     printf("Printing the list:\n");
18.     for (i = 0; i < size; i++)
19.     {
20.         list[i] = rand() % size;
21.         printf("%d    ", list[i]);
22.     }
23.     printf("\nEnter the key to find it's occurence: ");
24.     scanf("%d", &key);
25.     occur(list, size, 0, key, &count);
26.     printf("%d occurs for %d times.\n", key, count);
27.     return 0;
28.}
29.
30.void occur(int list[], int size, int index, int key, int *count)
31.{
32.     if (size == index)
33.     {
34.         return;
35.     }
36.     if (list[index] == key)
37.     {
38.         *count += 1;
39.     }
40.     occur(list, size, index + 1, key, count);
```

```
41.}
```

C Program Find the Length of the Linked List using Recursion

This C Program uses recursive function & calculates the length of a string. The user enters a string to find it's length.
Here is the source code of the C program to find the length of a string. The C Program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2.  * C program to find the length of a string
3.  */
4. #include <stdio.h>
5.
6. int length(char [], int);
7. int main()
8. {
9.       char word[20];
10.       int count;
11.
12.       printf("Enter a word to count it's length: ");
13.       scanf("%s", word);
14.       count = length(word, 0);
15.       printf("The number of characters in %s are %d.\n", word, count);
16.       return 0;
17.}
18.
19.int length(char word[], int index)
20.{
21.       if (word[index] == '\0')
22.       {
23.             return 0;
24.       }
25.       return (1 + length(word, index + 1));
26.}
```

C Program to find Product of 2 Numbers using Recursion

This C program using recursion, finds the product of 2 numbers without using the multiplication operator.
Here is the source code of the C program to display a linked list in reverse. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
```

```
 * C Program to find Product of 2 Numbers using Recursion
 */
#include <stdio.h>

int product(int, int);

int main()
{
    int a, b, result;

    printf("Enter two numbers to find their product: ");
    scanf("%d%d", &a, &b);
    result = product(a, b);
    printf("Product of %d and %d is %d\n", a, b, result);
    return 0;
}

int product(int a, int b)
{
    if (a < b)
    {
        return product(b, a);
    }
    else if (b != 0)
    {
        return (a + product(a, b - 1));
    }
    else
    {
        return 0;
    }
}
```

C Program to Find whether a Number is Prime or Not using Recursion

The following C program, using recursion, finds whether the entered number is a prime number or not. A prime number is an integer that has no integral factor but itself and 1.

Here is the source code of the C program to find an element in a linked list. The C Program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to find whether a Number is Prime or Not using Recursion
 */
#include <stdio.h>

```

```c
int primeno(int, int);

int main()
{
    int num, check;
    printf("Enter a number: ");
    scanf("%d", &num);
    check = primeno(num, num / 2);
    if (check == 1)
    {
        printf("%d is a prime number\n", num);
    }
    else
    {
        printf("%d is not a prime number\n", num);
    }
    return 0;
}

int primeno(int num, int i)
{
    if (i == 1)
    {
        return 1;
    }
    else
    {
        if (num % i == 0)
        {
            return 0;
        }
        else
        {
            return primeno(num, i - 1);
        }
    }
}
```

C Program to Print Binary Equivalent of an Integer using Recursion

This C program, using recursion, finds the binary equivalent of a decimal number entered by the user. Decimal numbers are of base 10 while binary numbers are of base 2.

Here is the source code of the C program to display a linked list in reverse. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```c
/*
 * C Program to Print Binary Equivalent of an Integer using Recursion
 */
#include <stdio.h>

int binary_conversion(int);

int main()
{
    int num, bin;

    printf("Enter a decimal number: ");
    scanf("%d", &num);
    bin = binary_conversion(num);
    printf("The binary equivalent of %d is %d\n", num, bin);
}

int binary_conversion(int num)
{
    if (num == 0)
    {
        return 0;
    }
    else
    {
        return (num % 2) + 10 * binary_conversion(num / 2);
    }
}
```

C Program to Print the Alternate Nodes in a Linked List using Recursion

This C program, using recursion, displays the alternate nodes in a linked list.A linked list is an ordered set of data elements, each containing a link to its successor.
Here is the source code of the C program to display a linked list in reverse. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```c
/*
 * C Program to Print the Alternate Nodes in a Linked List using Recursion
 */
#include <stdio.h>
#include <stdlib.h>

struct node
{
    int a;
```

```
10.     struct node *next;
11.};
12.
13.void generate(struct node **);
14.void display(struct node *);
15.void delete(struct node **);
16.
17.int main()
18.{
19.     struct node *head = NULL;
20.
21.     generate(&head);
22.     printf("\nDisplaying the alternate nodes\n");
23.     display(head);
24.     delete(&head);
25.
26.     return 0;
27.}
28.
29.void display(struct node *head)
30.{
31.     static flag = 0;
32.     if(head != NULL)
33.     {
34.          if (!(flag % 2))
35.          {
36.               printf("%d  ", head->a);
37.          }
38.          flag++;
39.          display(head->next);
40.     }
41.}
42.
43.void generate(struct node **head)
44.{
45.     int num, i;
46.     struct node *temp;
47.
48.     printf("Enter length of list: ");
49.     scanf("%d", &num);
50.     for (i = num; i > 0; i--)
51.     {
52.          temp = (struct node *)malloc(sizeof(struct node));
53.          temp->a = i;
54.          if (*head == NULL)
55.          {
56.               *head = temp;
57.               (*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);
      }
}
```

C Program to Convert a Number Decimal System to Binary System using Recursion

The following C program using recursion finds a binary equivalent of a decimal number entered by the user. The user has to enter a decimal which has a base 10 and this program evaluates the binary equivalent of that decimal number with base 2.
Here is the source code of the C program to find the binary equivalent of the decimal number. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Convert a Number Decimal System to Binary System using Recursion
 */
#include <stdio.h>

int convert(int);

int main()
{
      int dec, bin;

      printf("Enter a decimal number: ");
      scanf("%d", &dec);
      bin = convert(dec);
      printf("The binary equivalent of %d is %d.\n", dec, bin);

      return 0;
}
```

```
19.
20.int convert(int dec)
21.{
22.     if (dec == 0)
23.     {
24.          return 0;
25.     }
26.     else
27.     {
28.          return (dec % 2 + 10 * convert(dec / 2));
29.     }
30.}
```

C Program to find the First Capital Letter in a String using Recursion

The following C program, using recursion, finds the first capital letter that exists in a string. We have included ctype.h in order to make use of “int isupper(char);” function that’s defined inside the ctype.h headerfile. The isupper finction returns 1 if the passed character is an uppercase and returns 0 is the passed character is a lowercase.

Here is the source code of the C program to find the first capital letter in a string using recursion. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2.  * C Program to find the first capital letter in a string using
3.  * Recursion
4.  */
5. #include <stdio.h>
6. #include <string.h>
7. #include <ctype.h>
8.
9. char caps_check(char *);
10.
11.int main()
12.{
13.     char string[20], letter;
14.
15.     printf("Enter a string to find it's first capital letter: ");
16.     scanf("%s", string);
17.     letter = caps_check(string);
18.     if (letter == 0)
19.     {
20.          printf("No capital letter is present in %s.\n", string);
21.     }
22.     else
23.     {
24.          printf("The first capital letter in %s is %c.\n", string, letter);    }
25.          return 0;
26.     }
```

```
27.     char caps_check(char *string)
28.     {
29.         static int i = 0;
30.         if (i < strlen(string))
31.         {
32.             if (isupper(string[i]))
33.             {
34.                 return string[i];
35.             }
36.             else
37.             {
38.                 i = i + 1;
39.                 return caps_check(string);
40.             }
41.         }
42.         else return 0;
43.     }
```

```
#include <stdio.h> #define MAX 100 void ArrayElement(int arr1[], int st, int l); int main() { int arr1[MAX]; int n, i; printf("\n\n Recursion : Print the array elements :\n"); printf("----------------------------------------\n"); printf(" Input the number of elements to be stored in the array :"); scanf("%d",&n); printf(" Input %d elements in the array :\n",n); for(i=0;i<n;i++) { printf(" element - %d : ",i); scanf("%d",&arr1[i]); } printf(" The elements in the array are : "); ArrayElement(arr1, 0, n);//call the function ArrayElement printf("\n\n"); return 0; } void ArrayElement(int arr1[], int st, int l) { if(st >= l) return; //Prints the current array element printf("%d ", arr1[st]); /* Recursively call ArrayElement to print next element in the array */ ArrayElement(arr1, st+1, l);//calling the function ArrayElement itself }
```