L2 : Basic OF Recursion Practice Questions

1-Tut : Power

Send Feedback

Write a program to find x to the power n (i.e. x^n). Take x and n from the user. You need to return the answer.

Do this recursively.

Input format :

Two integers x and n (separated by space)

Output Format :

x^n (i.e. x raise to the power n)

Constraints :

1 <= x <= 30

0 <= n <= 30

Sample Input 1 :

3 4

Sample Output 1 :

81

Sample Input 2 :

2 5

Sample Output 2 :

32

```
int power(int x, int n) {
  /* Don't write main().
     Don't read input, it is passed as a function argument.
     Return output and don't print it.
     Taking input and printing output is handled automatically.
  */
   if(n == 0){
      return 1;
   }
   int smallans = power(x,n-1);
   return x * smallans;
}
```

2-Tut : Print Numbers

Send Feedback

Given is the code to print numbers from 1 to n in increasing order recursively. But it contains a few bugs that you need to rectify such that all the test cases pass.

Input Format :

Integer n

Output Format :

Numbers from 1 to n (separated by space)

Constraints :

1 <= n <= 10000

Sample Input 1 :

6

Sample Output 1 :

1 2 3 4 5 6

Sample Input 2 :

4

Sample Output 2 :

1 2 3 4

```
1.  void print(int n){
2.      if(n == 1){
3.          cout << n << " ";
4.          return;
5.      }
6.      //cout << n << " ";
7.      print(n - 1);
8.      cout << n << " ";
9.  }
```

3-Tut : Number of Digits

Send Feedback

Given the code to find out and return the number of digits present in a number recursively. But it contains a few bugs that you need to rectify such that all the test cases should pass.

Input Format :

Integer n

Output Format :

Count of digits

Constraints :

1 <= n <= 10^6

Sample Input 1 :

156

Sample Output 1 :

3

Sample Input 2 :

7

Sample Output 2 :

1

```
1.  int count(int n){
2.      if(n < 10){
3.          return 1;
4.      }
5.      int smallAns = count(n / 10);
6.      return smallAns + 1;
7.  }
```

4-Tut : What is the output

Send Feedback

What will be the output of the following code ?

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

int func(int num){
    return func(num- 1);
}

int main() {
    int num = 5;
    int ans = func(num - 1);
    cout << ans;
}
```

Options

Compilation Error

Runtime Error

5

None of these

Correct Answer : B

Solution Description

Since the base case is missing in the code, therefore there will be infinite recursion calls and hence runtime error will occur.

5-Tut :What is the output

Send Feedback

What will be the output ?

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

void print(int n){
    if(n < 0){
        return;
```

```
    }
    cout << n << " ";
    print(n - 2);
}

int main() {
    int num = 5;
    print(num);
}
```

Options

Runtime error
5 4 3 2 1
5 3 1
None of these

Correct Answer : C

6-Tut : What is the output

Send Feedback
What will be the output of the following code ?

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

void print(int n){
    if(n < 0){
        return;
    }
    if(n == 0){
        cout << n << " ";
        return;
    }
    print(n --);
    cout << n << " ";
}

int main() {
    int num = 3;
    print(num);
}
```

Options

Runtime Error
3 2 1
3 3 3
0 1 2 3

Correct Answer : A

Solution Description

In function print when a recursion call is being made n-- is being passed as input. Here we are using a post decrement operator, so the argument passed to the next function call is n and not n - 1. Thus there will be infinite recursion calls and runtime error will come.

Recursion and Arrays

7-Tut : Sum of Array

Send Feedback

Given an array of length N, you need to find and return the sum of all elements of the array.

Do this recursively.

Input Format :

Line 1 : An Integer N i.e. size of array

Line 2 : N integers which are elements of the array, separated by spaces

Output Format : Sum

Constraints :

1 <= N <= 10^3

Sample Input 1 :

3

9 8 9

Sample Output 1 : 26

Sample Input 2 :

3

4 2 1

Sample Output 2 :

7

```
int sum(int input[], int n) {
  /* Don't write main(). Don't read input, it is passed as function argument. Return output and don't print it.  Taking input and printing output is handled automatically. */
    if(n == 1){
        return input[0];
    }

    int smallans = sum(input+1,n-1);
    return input[0] + smallans;
}
```

8-Tut : Check Number

Send Feedback

Given an array of length N and an integer x, you need to find if x is present in the array or not. Return true or false.

Do this recursively.

Input Format :

Line 1 : An Integer N i.e. size of array
Line 2 : N integers which are elements of the array, separated by spaces
Line 3 : Integer x

Output Format :

'true' or 'false'

Constraints :

1 <= N <= 10^3

Sample Input 1 :

3
9 8 10
8

Sample Output 1 :

true

Sample Input 2 :

3
9 8 10
2

Sample Output 2 :

false

```
1.  bool checkNumber(int input[], int size, int x) {
2.  
3.      if(size < 1){
4.          return false;
5.      }
6.  
7.      if(input[0] == x){
8.          return true;
9.      }
10. 
11.     return checkNumber(input+1,size-1,x);
12. }
```

9-Tut : First Index of Number

Send Feedback

Given an array of length N and an integer x, you need to find and return the first index of integer x present in the array. Return -1 if it is not present in the array.

First index means, the index of first occurrence of x in the input array.

Do this recursively. Indexing in the array starts from 0.

Input Format :

Line 1 : An Integer N i.e. size of array
Line 2 : N integers which are elements of the array, separated by spaces
Line 3 : Integer x

Output Format :

first index or -1

Constraints :

1 <= N <= 10^3

Sample Input :

4
9 8 10 8
8

Sample Output :

1

```
1.  int firstIndex(int input[], int size, int x) {
2.
3.      if(size < 1){
4.          return -1;
5.      }
6.
7.      if(input[0] == x){
8.          return 0;
9.      }
10.
11.     int smallans = firstIndex(input+1,size-1,x);
12.     if(smallans != -1){
13.         return 1+smallans;
14.     }
15.     else{
16.         return smallans;
17.     }
18. }
```

10-Tut : Last Index of Number

Send Feedback

Given an array of length N and an integer x, you need to find and return the last index of integer x present in the array. Return -1 if it is not present in the array.

Last index means - if x is present multiple times in the array, return the index at which x comes last in the array.

You should start traversing your array from 0, not from (N - 1).

Do this recursively. Indexing in the array starts from 0.

Input Format :

Line 1 : An Integer N i.e. size of array
Line 2 : N integers which are elements of the array, separated by spaces
Line 3 : Integer x

Output Format :

last index or -1

Constraints :

1 <= N <= 10^3

Sample Input :

4
9 8 10 8
8

Sample Output :

3

```
int lastIndex(int input[], int size, int x) {
 /* Don't write main().
    Don't read input, it is passed as function argument.
    Return output and don't print it.
    Taking input and printing output is handled automatically.
 */

   if(size < 1)
   {
     return -1;
   }


   int smallans = lastIndex(input+1,size-1,x);

   if(smallans == -1){
      if(input[0] == x){
         return 0;
      }
      else{
         return smallans;
      }
   }
   else
   {
     return 1+smallans;
   }

}
```

11-Tut : All Indices of Number

Send Feedback

Given an array of length N and an integer x, you need to find all the indexes where x is present in the input array. Save all the indexes in an array (in increasing order).

Do this recursively. Indexing in the array starts from 0.

Input Format :

Line 1 : An Integer N i.e. size of array
Line 2 : N integers which are elements of the array, separated by spaces
Line 3 : Integer x

Output Format :

indexes where x is present in the array (separated by space)

Constraints :

1 <= N <= 10^3

Sample Input :

5
9 8 10 8 8
8

Sample Output :

1 3 4

```
int allIndexes(int input[], int size, int x, int output[]) {
    if(size < 1){
        return 0;
    }
    int oparrsize = allIndexes(input+1,size-1,x,output);

    for(int i = 0; i < oparrsize; i++){
        output[i] = output[i]+1;
    }

    if(input[0] == x)
    {

        for(int i = oparrsize-1; i >= 0; i--){
            output[i+1] = output[i];
        }
        output[0] = 0;
        return 1+oparrsize;
    }
    else
    {
        return oparrsize;
    }
}
```