Lecture 16 Recursion

1. Is array sorted?
2. #include <iostream>
3. using namespace std;
4. bool isSorted(int arr[], int n)
5. {
6. // base case
7. if(n==0 or n==1)
8. {
9. return true;
10. }
11. if(arr[0]<arr[1] && isSorted(arr+1, n-1))
12. {
13. return true;
14. }
15. return false;
16. }
17. int main() {
18. int arr[] = {1,2,3,14,5};
19. int n = 5;
20. if(isSorted(arr, n))
21. {
22. cout<<"Sorted!";
23. }
24. else
25. {
26. cout<<"Unsorted";
27. }
28. return 0;
29. }

2. Linear Search

#include <iostream>

using namespace std;

int linear\_search(int arr[], int n, int key, int i)

{

    // base

    if(i==n)

    {

        return -1;

    }

    if(arr[i]==key)

    {

        return i;

    }

    return linear\_search(arr, n, key, i+1);

}

int main() {

    int arr[] = {1,3,5,4,6};

    int n = 5;

    int key = 7;

    cout<<linear\_search(arr, n, key, 0);

    return 0;

}

3. Tower of Hanoi

#include <iostream>

using namespace std;

void TOH(int n, char A, char B, char C)

{

    // A -> src, B -> helper, C -> dest

    // base case

    if(n==0)

    {

        return;

    }

    TOH(n-1, A, C, B);

    cout<<"Move "<<n<<" disk from "<<A<<" to "<<C<<endl;

    TOH(n-1, B, A, C);

}

int main() {

    int n;

    cin>>n;

    TOH(n, 'A', 'B', 'C');

    return 0;

}

4. [https://practice.geeksforgeeks.org/problems/print-pattern3549/1/?problemStatus=solved&page=1&category[]=Recursion&query=problemStatussolvedpage1category[]Recursion#](https://practice.geeksforgeeks.org/problems/print-pattern3549/1/?problemStatus=solved&page=1&category%5b%5d=Recursion&query=problemStatussolvedpage1category%5b%5dRecursion#)

class Solution{

void pat(vector<int>&res, int n)

{

res.push\_back(n); // 16, 11, 6, 1, -4

if(n==0 || n<0)

{

return;

}

pat(res, n-5);

res.push\_back(n); // 1, 6, 11, 16

}

public:

vector<int> pattern(int N){

// code here

vector<int> res;

pat(res, N);

return res;

}

};

5. <https://leetcode.com/problems/generate-parentheses/submissions/>

class Solution {

void parenthesis(int idx, int n, int open, int close, char ch[], vector<string>&res)

{

if(idx==2\*n)

{

ch[idx]='\0';

res.push\_back(ch);

return;

}

if(open < n)

{

ch[idx] = '(';

parenthesis(idx+1, n, open+1, close, ch, res);

}

if(close < open)

{

ch[idx] = ')';

parenthesis(idx+1, n, open, close+1, ch, res);

}

}

public:

vector<string> generateParenthesis(int n) {

vector<string> res;

char ch[1000];

parenthesis(0, n, 0, 0, ch, res);

return res;

}

};