CRACKING CODING INTERVIEWS

## COMPETITIVE PROGRAMMING

- GOAL (Convert input to output in efficient way)

- MACHINE/COMPUTER/AUTO-EVALUATION/ONLINE-COMPILER

- Write full program

* Reading Input
* Declaring Variables
* Processing Input
* Producing Output

## CODING INTERVIEW

- Function

* Generate all subsets of given set {1,2,3}
  + The set will have 2^3 subsets

1

1 2

1 3

1 2 3

2

2 3

3

{} // Empty subset

- Function Signature

*vector*<*vector*<int>> generateAllSubsets(*vector*<int> set)

int[] generateAllSubsets(int[] set, int size)

void generateAllSubsets(int[] set, int size) // If we only print.

## DATA TYPES (Primitive)

- int, char (INTEGRAL)

- float (FLOATING)

- boolean (BOOL)

USER DEFINED OR CUSTOM DATA TYPES

- struct

- class

- enum

class Interval {

public:

int startTime;

int endTime;

};

1. Defining the data type.

2. Way of declaring a variable of that data type.

Interval a, b;

int a[]3

char c[];

Interval i[];

3. Access members of this data type

Interval a;

int st = a.startTime;

## Static Array

int a[10];

## Dynamic Array

Things to know...

1. Declaring the dynamic array.
2. Adding elements to the dynamic array.
3. Getting the size of dynamic array.
4. Accessing the elements of the dynamic array.
5. Iterating over the dynamic array.

### C++

*vector*<int> a;

a.*push\_back*();

a.*size*();

int ele = a[0];

### JAVA

ArrayList<Integer> al = new ArrayList<>();

al.add(12);

al.size();

al.get(0);

### C#

*List*<int> li = new *List*<int>();

li.Add(12);

li.*Length*;

int ele = li[0];

### PYTHON

lis = []

lis.*append*(12)

l = *len*(lis)

ele = lis[0]

### In C:

void\* *malloc*(int bytes);

void\* *realloc*(void\* p, int newSize);

void memset(void\*p, int val, int sizeInBytes);

void *memmove*(...)

## How it is implemented internally?

class DynamicArray {

int\* arr;

int capacity = 2;

int size = 0;

void init() {

arr = new int[capacity];

}

public:

void init(int initialCap) {

capacity = initialCap;

init();

}

void push\_back(int x) {

if (size < capacity)

arr[size++] = x;

else{

capacity = capacity \* 2;

int\* temp = new int[capacity];

*memcpy*(temp, arr, size);

delete[] arr;

arr = temp;

}

}

int get(int index) {

return arr[index];

}

};

## Recursion:

- Function calling itself directly or indirectly

- One call.

- Multiple calls.

Example:

int fib(int n) {

if (n <= 1)

eturn n;

else

return fib(n - 1) + fib(n - 2);

}

### Trace the below code for x = 3 , n = 3

int getAnswer(int x, int n)

{

if (n == 0)

return 1;

int half = getAnswer(x, n / 2);

int halfSqr = half \* half;

if ((n & 1) == 1)

return halfSqr \* x;

return halfSqr;

}

## Trace the below code:

#include <iostream>

#include <string>

using namespace *std*;

int a[] = { 1,2,3 };

int an = 3;

void allSubSets(int idx, *string* currentSubset) {

if (idx == an) {

*cout* << "{" << currentSubset << " }\n";

return;

}

allSubSets(idx + 1, currentSubset + " " + *to\_string*(a[idx]));

allSubSets(idx + 1, currentSubset);

}

int main() {

allSubSets(0, "");

return 0;

}

## Trace the below code:

#include <iostream>

#include <string>

using namespace *std*;

*string* s = "abc";

void swap(int i, int j) {

char temp = s[i];

s[i] = s[j];

s[j] = temp;

}

void allPermutations(int idx) {

if (idx == s.*length*()) {

*cout* << s << "\n";

return;

}

for (int nextIdx = idx; nextIdx < s.*length*(); nextIdx++) {

swap(idx, nextIdx);

allPermutations(idx + 1);

swap(idx, nextIdx);

}

}

int main() {

allPermutations(0);

return 0;

}