IIT Bombay

Computer Programming

Dr. Deepak B Phatak
Dr. Supratik Chakraborty
Department of Computer Science and Engineering
IIT Bombay

Session: Recap of Function Calls and Parameter Passing

Overall Program Structure

IIT Bombay

```
#include <iostream>
using namespace std;
int myEncode(int q1Marks,int q2Marks);
int power(int base, int exponent);
int main() { …
  for ( … ) {  …
   cipher = myEncode(q1Marks, q2Marks);
  …}
…}
```

```
// PRECONDITION:  …
int myEncode(int q1Marks,
                    int q2Marks)
{ …
  twoRaisedQ1 = power(2, q1Marks);
  threeRaisedQ2 = power(3, q2Marks);
 … }
// POSTCONDITION:  …
```

```
// PRECONDITION:  …
int power(int base, int exponent)
{ … }
// POSTCONDITION:  …
```

Contract View of Functions

IIT Bombay

```
#include <iostream>
using namespace std;
int myEncode(int q1Marks, int q2Marks);
int main() {
    …
    for ( … ) {  …
      cipher = myEncode(q1Marks, q2Marks);
      …
    }
    …
}
```

Ensure pre-condition of “myEncode” before invoking

Guaranteed post-condition of “myEncode” on returning

```
// PRECONDITION:
// 1 <= q1Marks <= 10
// 1 <= q2Marks <= 10
int myEncode(int q1Marks,
                 int q2Marks)
{
       BLACK BOX
}
```

// POSTCONDITION:
//   Returned value =
// $2^{q1Marks} \times 3^{q2Marks}$
// No side effects (later lecture)

Flow of Control: An Animation

IIT Bombay

```
#include <iostream>
using namespace std;
int myEncode(int q1Marks,int q2Marks);
int power(int base, int exponent);
int main() { …
  for (…) { …
    ipher = myEncode(q1Marks, q2Marks);
  … }
  …
return 0; }
```

```
int myEncode(int q1Marks,
             int q2Marks)
{ …
  oRaisedQ1 = power(2, q1Marks);
  eeRaisedQ2 = power(3, q2Marks);
  …
  return cipher;
}
```

```
int power(int base, int exponent)
{ …
  return result;
}
```

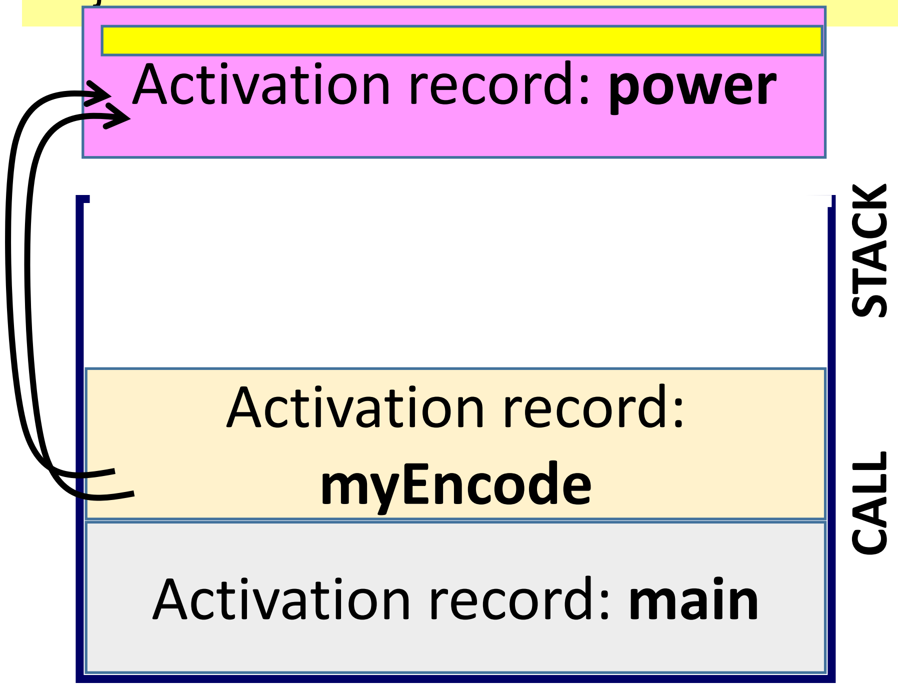
Activation Records in Call Stack

IIT Bombay

When a function (caller) calls a function (callee)

- a **fresh** activation record for callee created
- Values of function parameters from caller copied to space allocated for formal parameters of callee
- PC of caller saved
- Other book-keeping information updated
- Activation record for callee pushed on call stack

```
int
myEncode(int q1Marks, int q2Marks)
{ ….
  twoRaisedQ1 = power(2, q1Marks);
 …}
```



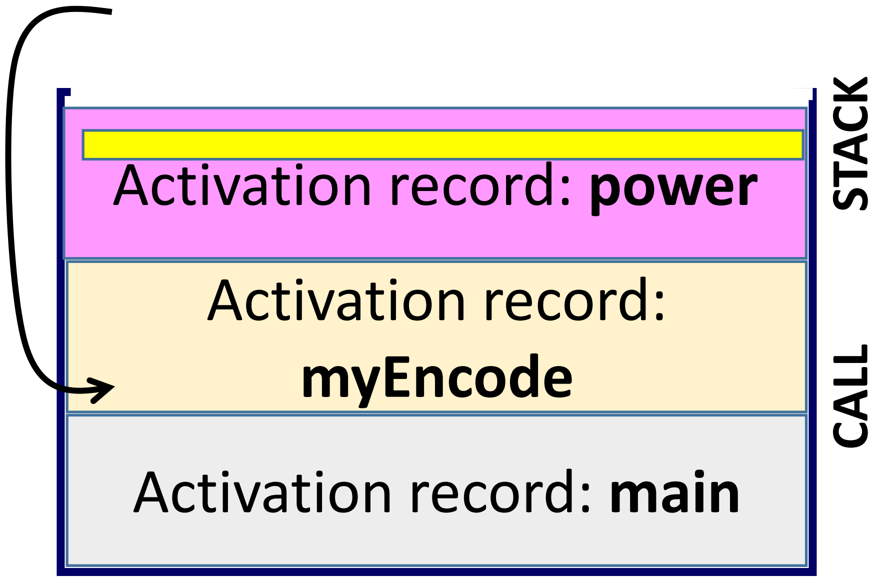
Activation Records in Call Stack

IIT Bombay

When a function (callee) returns

- Callee's activation record popped from call stack
- Return value from popped activation record copied to activation record of caller (now on top of stack)
- Value of PC saved in popped activation record loaded in PC of CPU
- Free activation record of callee
- Resume execution of instruction at location given by updated PC

```
int power(int base, int exponent)
{ ....
  return result;
 ...}
```



Call-by-Value Parameter Passing

IIT Bombay

Values of function parameters copied from activation record of caller to activation record of callee

Recall:

Formal parameters of callee (**power**) are its local variables

Not confused with parameters used in caller (**myEncode**) when invoking callee (**power**)

Only way in which callee (**power**) can let caller (**myEncode**) see effects of its computation is through return value of callee

Call-by-Reference Parameter Passing

IIT Bombay

```
#include <iostream>
using namespace std;
int swap(int &m, int &n);
int main() {
```

m and n are NOT local variables of swap, but **references** (or **aliases**) to caller variables (a and b) used to pass parameters

```
  return 0;
}
```

```
int swap(int &m, int &n)
{
  int temp;
  temp = m;
  m = n;
  n = temp;
  return 0;
}
```

Practice Problem 1

IIT Bombay

Write a C++ function intSqRoot that takes a non-negative double input parameter and returns a double value such that

*** fractional part of intSqRoot(x) is always 0.0**

*** intSqRoot(x) is always non-negative**

*** $(\text{intSqRoot}(x))^2 <= x < (\text{intSqRoot}(x) + 1)^2$**

Practice Problem 1 (continued)


```
double intSqRoot(double x) {
  double result;
   // Input validation

   // Your code to compute intSqRoot
  return result;
}
```

Practice Problem 2

IIT Bombay

We want to write a program that takes the co-ordinates of 3 points, each in 3-dimensional space, and finds which one of them is farthest from the origin, and which is nearest to the origin.

The program then prints the nearest and farthest points, along with the integral part of their distances from the origin.

Practice Problem 2 (continued)

IIT Bombay

Each point is represented by a triple of floating-point coordinates (x, y, z).

The distance of the point (x, y, z) from the origin is the positive square root of

$$x^2 + y^2 + z^2$$

Practice Problem 2 (continued)

IIT Bombay

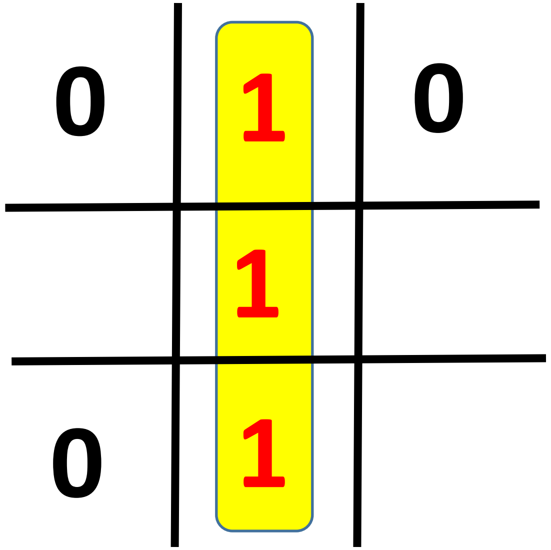
Write a C++ program to solve the above problem. Use the intSqRoot function designed in Practice Problem 1.

Optional: How would you extend your program to accept "n" points and find the farthest and nearest ones, where "n" is user provided. [You cannot use arrays].

Practice Problem 3

IIT Bombay

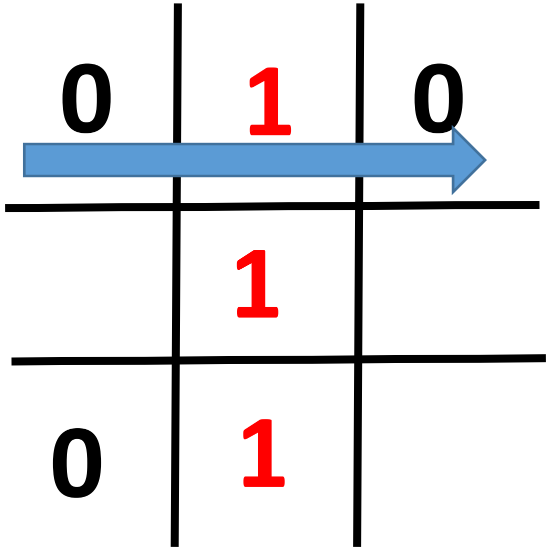
We want to write a C++ function that can be eventually used to play a game of tic-tac-toe (using 0's and 1's).



Practice Problem 3 (continued)

IIT Bombay

A configuration of the tic-tac-toe grid is represented by a sequence of 9 integers

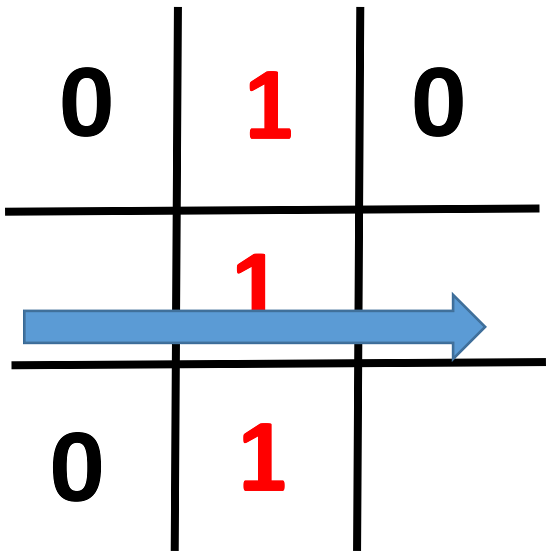


0, 1, 0,

Practice Problem 3 (continued)

IIT Bombay

A configuration of the tic-tac-toe grid is represented by a sequence of 9 integers

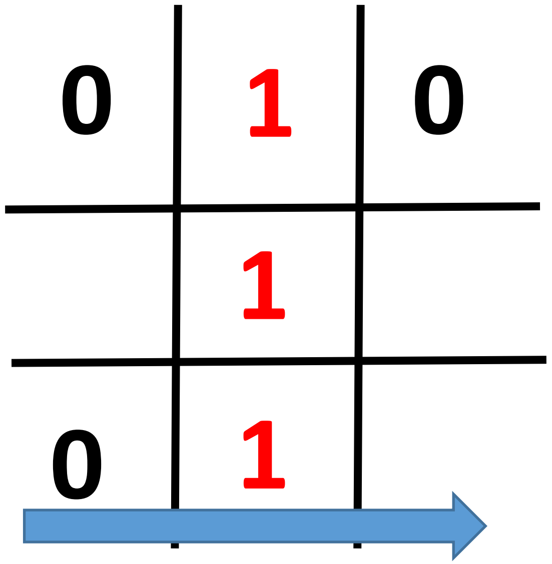


0, 1, 0, -1, 1, -1,

Practice Problem 3 (continued)

IIT Bombay

A configuration of the tic-tac-toe grid is represented by a sequence of 9 integers



0, 1, 0, -1, 1, -1, 0, 1, -1

Practice Problem 3 (continued)

IIT Bombay

A valid configuration has number of "0"s either equal to number of "1"s or more by one. Similarly, a valid configuration can have at most one winning line of "0"s or "1"s.

Practice Problem 3 (continued)

IIT Bombay

Example of invalid configuration

0	1	0
1	1	1
0	1	

0, 1, 0, 1, 1, 1, 0, 1, -1

Practice Problem 3 (continued)

IIT Bombay

Write a C++ function “tttCheckConfig” that checks if a configuration of the game is a valid one.

Your function should return the boolean value true if the configuration is valid, else it should return the boolean value false.

Practice Problem 4

IIT Bombay

Write a C++ function "tttReferee" that takes as inputs a sequence of 9 integers in {-1, 0, 1} representing a config of tic-tac-toe, and returns

1 if "1" has a winning config,

0 if "0" has a winning config,

2 otherwise

Practice Problem 5

IIT Bombay

Using the functions “tttCheckConfig” and “tttReferee”, we want to write a C++ program that plays a game of tic-tac-toe with the user. The user always plays first and uses “0”. The program and the user alternate with their turns, and the program uses “1”.

Practice Problem 5 (continued)

IIT Bombay

Positions on tic-tac-toe grid

1	2	3
4	5	6
7	8	9

Practice Problem 5 (continued)

IIT Bombay

The user indicates her choice of position for the next “0” by providing the position of the tic-tac-toe grid.

The program must read in this input, and find a position for the next “1” such that we have a valid configuration, and (hopefully, a winning configuration for “1”).

It then outputs the position of “1” so that the user can read it. This process continues until either the grid is filled or somebody wins.

Practice Problem 5 (continued)

IIT Bombay

Typical run of your program:

Position of 0: 3 (user input)

Position of 1 is 5 (program output)

Position of 0: 6 (user input)

Position of 1 is 1 (program output)

Position of 0: 9 (user input)

0 is winner

1 1	2	3 0
4	5 1	6 0
7	8	9 0