**What is a stack** –

* A stack is a special region in the computer memory that stores temporary variables created by each function (including the main() function).
* The stack is a LIFO (last in first out) data structure.
* Every time a function declares a new variable, it is pushed inside the stack.
* Every time a function exits, all the variables (that are already in the stack) are freed and removed from the stack.
* Stack variables are **local** in nature since when the function exits, the variables are popped out of the stack and are lost forever.
* That’s why attempting to access variable that was declared in a function is a bug (since the variable was popped out of the stack and does not exist anymore).

Stack advantages:

* The memory is managed for you. You don’t need to allocate memory for the variables (and you don’t need to free the variables once you don’t need them anymore).
* The CPU organizes the stack memory very efficiently so reading and writing stack variables is very fast.

Stack disadvanteges:

* (not sure it’s a disadvantage) there’s a limit on a variable size that is put in a stack. A too big variable cannot be put in the stack (This is NOT the case for a **heap**).

**What is a heap:**

* The heap is a region on the computer memory that is NOT managed automatically for you.
* The heap is NOT tightly managed by the CPU.
* The heap is larger than stack
* The heap is more free-floating (?) region of the memory.
* To allocate memory on the heap you must use **malloc** or **calloc**.
* The memory is deallocated using **free**.
* If you fail do that the program will have **memory leak**.
* **Unlike the stack**, the heap does not have size restriction on variables sizes.
* **Unlike the stack** the variables on the heap are accessible from outside the function. They can be read from anywhere on the program.
* Heap variables are essentially **global** in scope.

heap disadvantages:

* **Unlike the stack**, Heap memory is slightly slower to write to and read from.

When to use a heap:

* If you want to use a large array that you want to use for a long time (like global array) then you should keep it on a **heap**.
* If you are dealing with realtively small variables that only need to persist as long as the function using them is alive, then you should use the **stack**.

**Q0**) almost all companies :(

Why University of Haifa and not Technion?

=====================================================

**Q1**) Intel (10-15~ minutes) 2018  
Two threads are running the bellow code,   
what will be [min, max]   
values of x? can give such a sequence?  
  
shared int x = 0;     
for (int i=0; i<100; i++)  
{  
     x++;  
}

=====================================================  
  
**Q2**) Intel (1 hour~) 2018  
  
Implement printf from scratch

=====================================================

**Q3**)  2018  
what will the output of this program be?  
Now 3 different processes are printing to their own window. Will the

Output be the same?

while(1)

{

    printf(“%d\n”, malloc(1024));

}

=====================================================

**Q4**) Intel 2018

Implement your own aligned malloc

===================================================

**Q5**) Intel 2018

Explain what context switch is, and what is happening behind the scenes.

=====================================================

**Q6**) Intel 2018

Given 100 programs, each needs 100GB of memory, and the RAM is only 100GB. How is it possible they are all active and running?

Explain to details!

Supoosed each program is for a different process.

Each process needs to use 100GB of memory.

The process in most cases doesn’t use all the 100GB at once. It uses only small part of it at specific time. The rest of the data (that is not currently used) is on the disk.

The RAM (which is physical memory) is 100GB. Meaning, the process can access this memory immediately. But since the process doesn’t really use all the 100GB at once, it access a small part of it immediately and when it needs data that is not on the RAM, it brings it from the disk. This is called paging. Or swap in (swap out is from the physical memory to the disk).

**Q7**) Intel 2016

After a quick explanation of the rules, explain and write a pseudo-code of a calculator that converts Roman numbers into decimal numbers and vice-versa.

=====================================================

**Q8**) Marvell 2016

You are given an array of objects in three colors: red, blue, green in some random order.

Write a pseudocode of a function that re-arranges the array such that all the red objects appear first, then all the blue ones, then the green.

=====================================================

**Q9**) Intel 2018

Given two linked lists, one of size N and the other one of size M.

1. Find if they have a common node
2. Find this node in linear time

=====================================================

**Q10**) 2018

Given a linked list, the final node is pointing back to some node making a loop.

Find that noob in linear time.

You are not allowed to change the struct adding flags and so on…

(struct contains 2 fields, int i; and a pointer next)

=====================================================

What will be the output of this program?

while(1)

printf("%d\n",malloc(1024));

prints the pointer of the alollocated memory.

supposed malloc returns the address 1000 (it returns it as void\*).

then printf prints "1000".

malloc returns in page sizes (for example 4KB).

meaning that if I requested 1024 bytes it will return a page of size 4096 bytes in which ALL THE BYTES are available for use.

Will the output of this program be the same for 3 different processes?

-----------------------------------------------------------------------------

Write a function that returns 1 if stack grows up and 0 if stack grows down.

-----------------------------------------------------------------------------

What is the job of the BIOS? What do you think is its size? Where would it be located?

-----------------------------------------------------------------------------

What is the bootloader, what do you think its size is?

-----------------------------------------------------------------------------

Is the whole kernel loaded into memory while starting the computer?

-----------------------------------------------------------------------------

What is the range value of x? (2 threads running same program and x is shared). Give execution sequence for max and min.

shared int x=0;

for (int i=0;i<100; i++)

x++;

-----------------------------------------------------------------------------

Given 100 programs, each of size 100 GB, and RAM size is 100GB, how can this be possible?

-----------------------------------------------------------------------------

Given 2 programs, A and B, A runs infinite loop, and there's only 1 CPU in the system.

Will program B run?

-----------------------------------------------------------------------------

What is context switch?

-----------------------------------------------------------------------------

What is page fault? what is interrupt? what is the name of the interrupt triggered upon a page fault?