Stack vs Heap Memory Allocation

Memory in a C/C++ program can either be allocated on stack or heap.

Prerequisite : Memory layout of C program.

Stack Allocation : The allocation happens on contiguous blocks of memory. We call it stack memory allocation because the allocation happens in function call stack. The size of memory to be allocated is known to compiler and whenever a function is called, its variables get memory allocated on the stack. And whenever the function call is over, the memory for the variables is deallocated. This all happens using some predefined routines in compiler. Programmer does not have to worry about memory allocation and deallocation of stack variables.

Heap Allocation : The memory is allocated during execution of instructions written by programmers. Note that the name heap has nothing to do with heap data structure. It is called heap because it is a pile of memory space available to programmers to allocated and de-allocate. If a programmer does not handle this memory well, memory leak can happen in the program.

Key Differences Between Stack and Heap Allocations

1. In a stack, the allocation and deallocation is automatically done by whereas, in heap, it needs to be done by the programmer manually.
2. Handling of Heap frame is costlier than handling of stack frame.
3. Memory shortage problem is more likely to happen in stack whereas the main issue in heap memory is fragmentation.
4. Stack frame access is easier than the heap frame as the stack have small region of memory and is cache friendly, but in case of heap frames which are dispersed throughout the memory so it cause more cache misses.
5. Stack is not flexible, the memory size allotted cannot be changed whereas a heap is flexible, and the allotted memory can be altered.
6. Accessing time of heap takes is more than a stack.

Comparison Chart:

Parameter	STACK	HEAP
Basic	Memory is allocated in a contiguous block.	Memory is allocated in any random order.
Allocation and Deallocation	Automatic by compiler instructions.	Manual by programmer.
Cost	Less	More
Implementation	Hard	Easy
Access time	Faster	Slower
Main Issue	Shortage of memory	Memory fragmentation
Locality of reference	Excellent	Adequate
Flexibility	Fixed size	Resizing is possible