forNextDay() 3

The description of the four regions of the process memory

There are four regions of the process memory: the first is the text, the second is the data, the third is the heap, and the fourth is the stack – also known as the call stack. The stack, heap, and program (text) memories are volatile, meaning they require electricity and can keep data only when a computer is turned on. The text is the executable program. It is the place in which the content of the binary executable is loaded into. The text is located at the top of the process memory. Beneath the text is the data. The data contains memory available from anywhere within the program (global variables and static variables). The heap is beneath the data. It is responsible for run-time memory. In other words, it is responsible for accommodating the need for programs to request memory during execution, for example the Java keyword *new*. The fourth process memory region is the stack. It is beneath the heap and at the bottom of the process memory. This is where local variables, frames of functions (stack frames), return lines, and value addresses are stored. In the stack, each function of a program has its own frame. A new frame is generated when a function is called. If a function is called more than one time, each call has a corresponding return line.

How the stack memory works

A stack (call stack) works in the order of last in first out or first in last out. A stack stores frames, symbols, addresses, and values. A stack frame is formed when a function is called. After a function is called the return location is pushed in the frame of the called function, for instance if program A calls program B, the return location is pushed in the frame of program B. The return location is always the first thing pushed in the frame of a called function. If a function is called more than once, each call has its corresponding return location. If a variable is to store the return value of a called function, the variable would be given “garbage” value in the beginning. After the called function returns its value, the value of the variable will be modified from “garbage” to the return value. The address of such variable (one to store the return value of a called function) is called the value address. The value address is pushed in the stack frame of the called function directly above the return location. After the value address is pushed, if a called function has arguments, they are pushed in the stack frame of the called function left to right, meaning the left most variable is pushed is pushed first and then the next and so on until all the arguments are pushed. After the arguments are pushed, the local variables of the called function are pushed top to bottom, meaning the top local variable is pushed first and then the next and so on until they are all pushed. Also, an array is pushed on the stack element by element. The first pushed array

element is the one at index 0, then the element at index 1, after that the one at index 2, and so on until the array elements are all pushed.

Stack Frame of the given program

Frame	Symbol	Address	Value
	iArr[3]	0xfff9	garbage
	iArr[2]	0xfffa	garbage
	iArr[1]	0xfffb	garbage
main	iArr[0]	0xfffc	garbage
	d	0xfffd	garbage
	i	0xfffe	garbage
	c	0xffff	garbage