Question 1: figuring out where each line is located(heap,stack,etc..) using nm,size,objdump and gdb

Line number in code	nm	objdump	size
5	0000000000bc5060 B globBuf	0000000000bc5060 g O .bss 0000000000010000 globBuf	65536
6	0000000000201010 D primes	0000000000201010 g O .data 0000000000000010 primes	16
33	0000000000201020 d key.2775	0000000000201020 l O .data 0000000000000004 key.2775	4
34	0000000000201060 b mbuf.2776	0000000000201060 l O .bss 00000000009c4000 mbuf.2776	10240000
35	I couldn't locate where char ptr is located so I followed the conventions, which say that pointer is located on the stack.		

Important notice: The symbol in nm shows which part of memory layout it belongs(in objdump it is written directly B stands for bss etc..).
Using GDB(gnu debugging tool) is optimal for checking variable functions located on the stack.
After debugging I print the variables and function calls located on stack.

```
(gdb) b main
Breakpoint 1 at 0x711: file src_code.c, line 38.
(gdb) run
Starting program: /home/ron/Desktop/final_assignment/a.out

Breakpoint 1, main (argc=1, argv=0x7fffffffdfa8) at src_code.c:38
warning: Source file is more recent than executable.
38          doCalc(key);
(gdb) step
doCalc (val=9973) at src_code.c:20
20          printf("The square of %d is %d\n", val, square(val));
(gdb) step
square (x=9973) at src_code.c:13
13          result = x * x;
(gdb) step
14          return result;              /* 5. How the return value is passed? */
(gdb) step
15      }
(gdb) info stack
#0  square (x=9973) at src_code.c:15
#1  0x00005555555546b5 in doCalc (val=9973) at src_code.c:20
#2  0x00005555555471e in main (argc=1, argv=0x7fffffffdfa8) at src_code.c:38
(gdb) info locals
result = 99460729
```

Square,docalc,main are functions located on stack.

```
static int
square(int x)                   /* 3.stack frame */
{
 68a:   55                      push   %rbp
 68b:   48 89 e5                mov    %rsp,%rbp
 68e:   89 7d ec                mov    %edi,-0x14(%rbp)
    int result;                 /* 4. Where is allocated? */

    result = x * x;
 691:   8b 45 ec                mov    -0x14(%rbp),%eax
 694:   0f af 45 ec             imul   -0x14(%rbp),%eax
 698:   89 45 fc                mov    %eax,-0x4(%rbp)
    return result;              /* 5. How the return value is passed? */
 69b:   8b 45 fc                mov    -0x4(%rbp),%eax
}
 69e:   5d                      pop    %rbp
 69f:   c3                      retq
```

By examining the square function we can see the value is returned via register %rbp(tool used objdump).

```
        int t;                  /* 7. Where is allocated? */

        t = val * val * val;
 6d6:   8b 45 ec                mov    -0x14(%rbp),%eax
 6d9:   0f af 45 ec             imul   -0x14(%rbp),%eax
 6dd:   8b 55 ec                mov    -0x14(%rbp),%edx
 6e0:   0f af c2                imul   %edx,%eax
 6e3:   89 45 fc                mov    %eax,-0x4(%rbp)
```

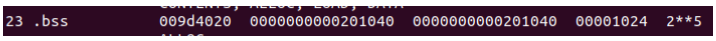
Rbp-register base pointer(points to start of stack) this we get by the tool objdump(like the picture above).

Size tool:

```
ron@ron-VirtualBox:~/Desktop/final_assignment$ size src_code
   text    data     bss     dec     hex filename
   1829     628 10305568        10308025         9d49b9 src_code
```

bss=globBuf size+mbuf size+extra 32 bit chunk= 65536+10240000+32=10305568

Note for the reader: Using the objdump tool we can where the extra 32 bits are coming from:



2^5=32(align column).

bss+data+text=10308025.
Like bss, data and text sections takes extra bits==>this can be examined by running tool ‘size’ on an executable with just a function main(each section has size and is non zero).