Q1:
When compiled and run this program crashes with a segmentation fault. This is because it is trying to access NULL in memory which is not a valid address and is therefore outside the valid memory segment allocated to this process.

Q2:
GDB shows where the fault occurs, the signal from the operating system, SIGSEGV on my particular system, and the code which caused the fault with some highlighting to show what is causing the fault. This gives a little bit more granular description of the problem by giving the system flag and the exact area where it believes that the fault is occurring.

Q3:
Valgrind shows where and what caused the fault, as well as in this case what memory addressed was attempted to be accessed and the fact that it has not been created or discard recently. In addition it shows that the heap was not used in this program giving a good snapshot of what memory looked like at the time of the fault.

Q4:
When run on its own q4 appears to have no problems and runs to completion with no errors. The same behavior is shown when run in GDB, the program is run and exits normally. When run with valgrind we can see that 4 bytes are being used from the heap that are then never freed, as seen in the heap usage summary. In the leak summary it shows 4 bytes as definitely lost. When run again through valgrind with the leak-check flag set to full we see that the leak is caused by a malloc call in main within the file q4 and the memory addresses at which the process was stored when it ran.

Q5:
When run on its own q5 runs with no errors. When run with valgrind we see that 400 bytes of memory have definitely been lost in 1 block. This is correct as the memory is allocated but never freed and since it was allocated all at once in one call to malloc it should be in one large block. Also as seen in Q4 one integer is 4 bytes and so 100 of the should be 400 bytes as seen here.

Q6:
When run on its own when a value is displaced from the freed array, the program does run and complete without errors. However the value that is displayed is a random trash value and not the value that was put into that cell of the array. The expected value was 1 but the value printed on this particularly run is 1455277551. This value does change with every subsequent run of the program. When run through valgrind we see that we have no memory leaks, but we have an invalid read of size 4 bytes, or the size of 1 integer. We can see that it occurred in main and that it was an address from withing a block of memory that was recently freed as well as the addresses associations with the call and the block.

Q7:
When compiled with an attempt to free a pointed from within the middle of the array the compiler gives a warning that free was called on a pointed with a non-zero offset. This occurs with several methods of passing the pointer to free. &data[50] and data+50 were both used in this test. When run without regard of the warning received the program crashed with the error of free() : invalid pointed.