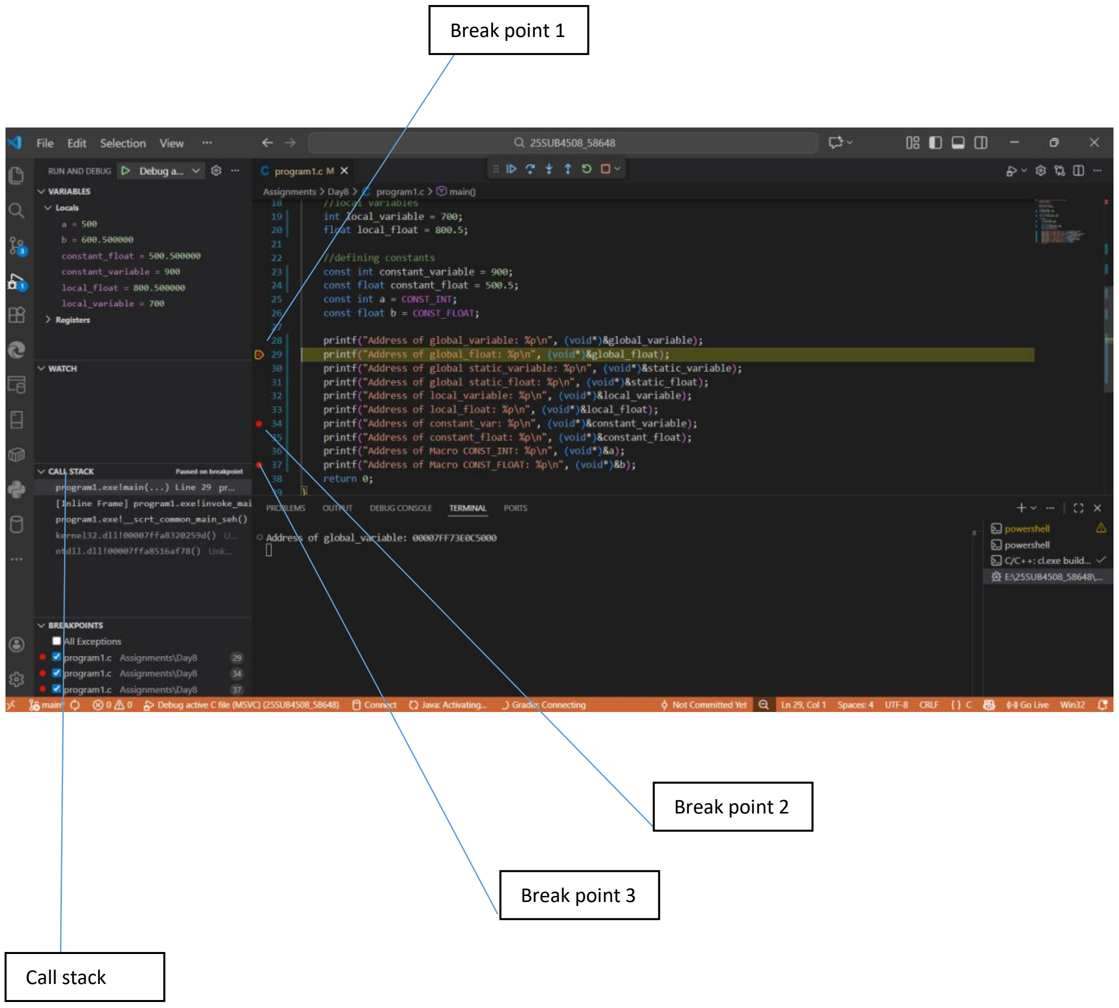
Task-1A

Observe layout of data in memory using gdb

Define variables of different types initialized with predefined data. The variables should be local variables, global variables, static variables, constants. Put a break point in gdb on main function and observe the memory layout of the variables in memory. Understand stack and heap memory areas.

```c
// observe layout of data in memory using gdb
#include <stdio.h>
#include <stdlib.h>

//defining constants
#define CONST_INT 500
#define CONST_FLOAT 600.5

//global variables
int global_variable = 100;
float global_float = 200.5;

//static variables
static int static_variable = 300;
static float static_float = 400.5;

int main() {
    //local variables
    int local_variable = 700;
    float local_float = 800.5;

    //defining constants
    const int constant_variable = 900;
    const float constant_float = 500.5;
    const int a = CONST_INT;
    const float b = CONST_FLOAT;

    printf("Address of global_variable: %p\n",
(void*)&global_variable);
    printf("Address of global_float: %p\n", (void*)&global_float);
    printf("Address of global static_variable: %p\n",
(void*)&static_variable);
    printf("Address of global static_float: %p\n",
(void*)&static_float);
    printf("Address of local_variable: %p\n", (void*)&local_variable);
    printf("Address of local_float: %p\n", (void*)&local_float);
    printf("Address of constant_var: %p\n", (void*)&constant_variable);
    printf("Address of constant_float: %p\n", (void*)&constant_float);
    printf("Address of Macro CONST_INT: %p\n", (void*)&a);
    printf("Address of Macro CONST_FLOAT: %p\n", (void*)&b);
    return 0;
}
```



```
a = 500
b = 600.500000
constant_float = 500.500000
constant_variable = 900
local_float = 800.500000
local_variable = 700
```

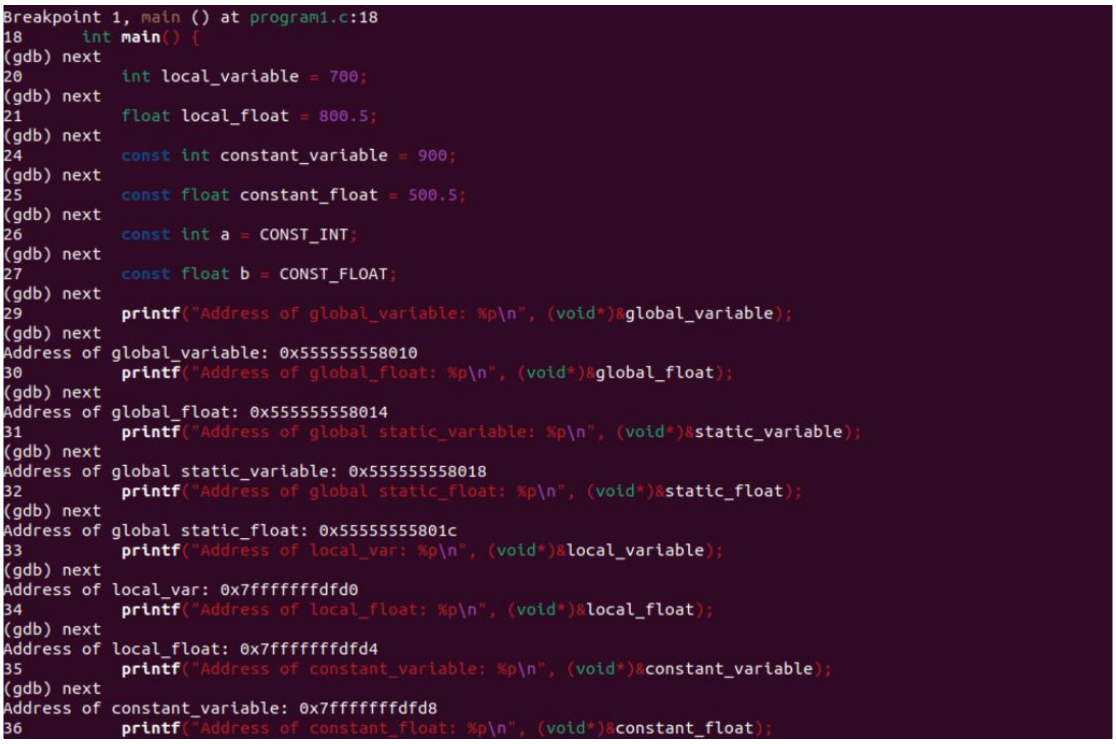
```
program1.exe!main(...) Line 29
[Inline Frame] program1.exe!invoke_mai
program1.exe!__scrt_common_main_seh()
kernel32.dll!00007ffa8320259d()
ntdll.dll!00007ffa8516af78()
```

```
program1.c  Assignments\Day8  29
program1.c  Assignments\Day8  34
program1.c  Assignments\Day8  37
```

```c
    //local variables
    int local_variable = 700;
    float local_float = 800.5;

    //defining constants
    const int constant_variable = 900;
    const float constant_float = 500.5;
    const int a = CONST_INT;
    const float b = CONST_FLOAT;

    printf("Address of global_variable: %p\n", (void*)&global_variable);
    printf("Address of global_float: %p\n", (void*)&global_float);
    printf("Address of global static_variable: %p\n", (void*)&static_variable);
    printf("Address of global static_float: %p\n", (void*)&static_float);
    printf("Address of local_variable: %p\n", (void*)&local_variable);
    printf("Address of local_float: %p\n", (void*)&local_float);
    printf("Address of constant_var: %p\n", (void*)&constant_variable);
    printf("Address of constant_float: %p\n", (void*)&constant_float);
    printf("Address of Macro CONST_INT: %p\n", (void*)&a);
    printf("Address of Macro CONST_FLOAT: %p\n", (void*)&b);
    return 0;
```

```
Address of global_variable: 00007FF73E0C5000
```

Using gdb



```
Breakpoint 1, main () at program1.c:18
18      int main() {
(gdb) next
20          int local_variable = 700;
(gdb) next
21          float local_float = 800.5;
(gdb) next
24          const int constant_variable = 900;
(gdb) next
25          const float constant_float = 500.5;
(gdb) next
26          const int a = CONST_INT;
(gdb) next
27          const float b = CONST_FLOAT;
(gdb) next
29          printf("Address of global_variable: %p\n", (void*)&global_variable);
(gdb) next
Address of global_variable: 0x555555558010
30          printf("Address of global_float: %p\n", (void*)&global_float);
(gdb) next
Address of global_float: 0x555555558014
31          printf("Address of global static_variable: %p\n", (void*)&static_variable);
(gdb) next
Address of global static_variable: 0x555555558018
32          printf("Address of global static_float: %p\n", (void*)&static_float);
(gdb) next
Address of global static_float: 0x55555555801c
33          printf("Address of local_var: %p\n", (void*)&local_variable);
(gdb) next
Address of local_var: 0x7fffffffdfd0
34          printf("Address of local_float: %p\n", (void*)&local_float);
(gdb) next
Address of local_float: 0x7fffffffdfd4
35          printf("Address of constant_variable: %p\n", (void*)&constant_variable);
(gdb) next
Address of constant_variable: 0x7fffffffdfd8
36          printf("Address of constant_float: %p\n", (void*)&constant_float);
```

Call stack:

- call stack keep track of function calls and variable we use .call stack is a stack follows LIFO order to provide faster access.
- Data exist here until function execution feed freed here.
- Automatic Memory allocation
- Exception –StackOverFlow error

Heap Memory:

- Heap memory follows first in first out (FIFO).
- Manual memory allocation, we need to free it.
- Slower accesss due to dynamic allocation
- Execution – OutofMemoryError