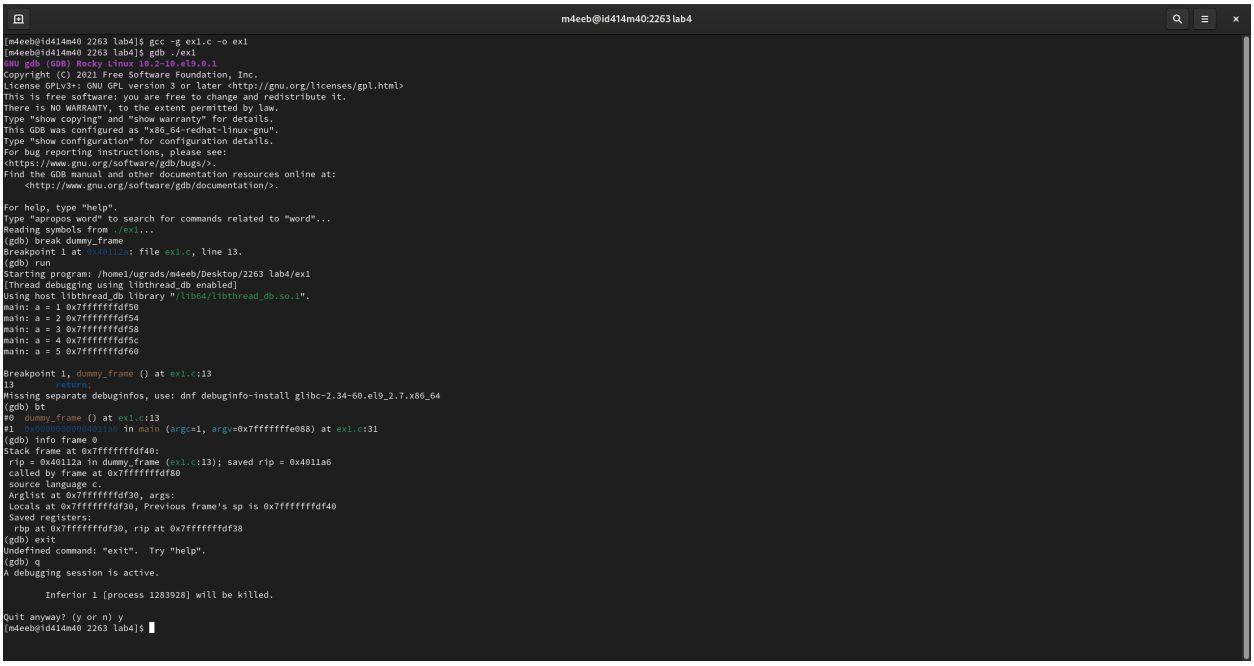
Exercise 1

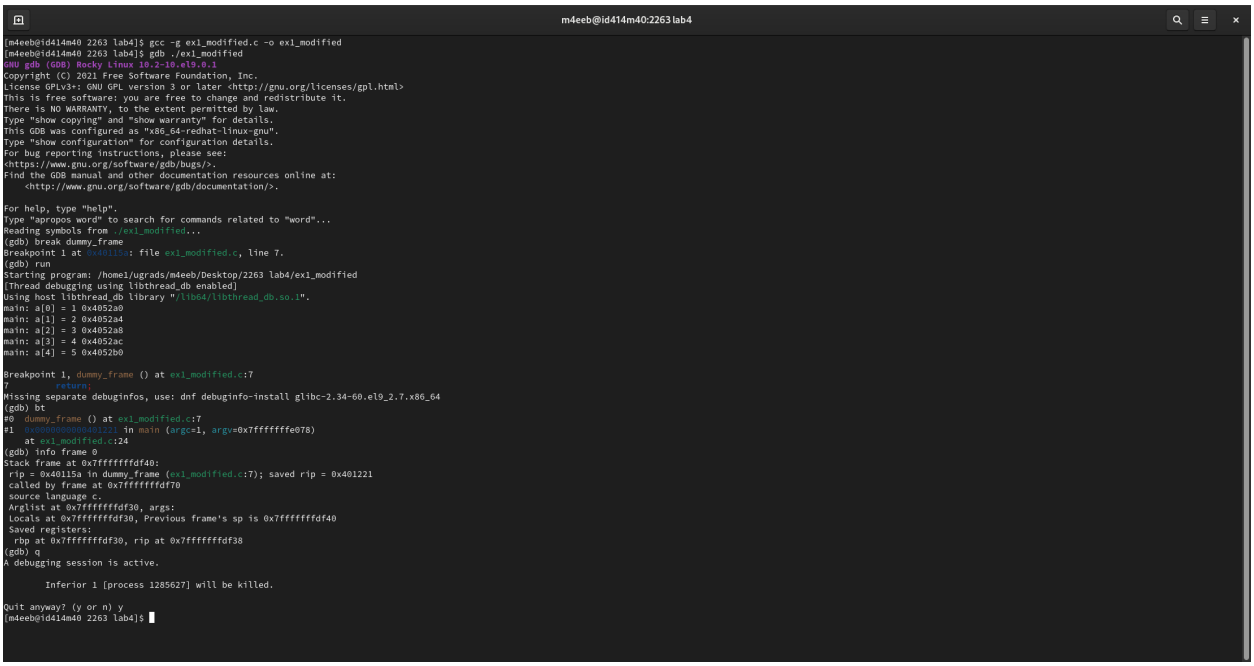


1.2

```
#include <stdio.h>
#include <stdlib.h>

void dummy_frame()
{
  return;
}

int main(int argc, char * * argv)
{
  int i;
  int *a = (int *)malloc(5 * sizeof(int));
  if (a == NULL) {
    fprintf(stderr, "Memory allocation failed\n");
    return EXIT_FAILURE;
  }

  for (i = 0; i < 5; i++) {
    a[i] = i + 1;
    printf("main: a[%d] = %d %p \n", i, a[i], &a[i]);
  }
```

1

```
  dummy_frame();

  free(a);
  return EXIT_SUCCESS;
}
```

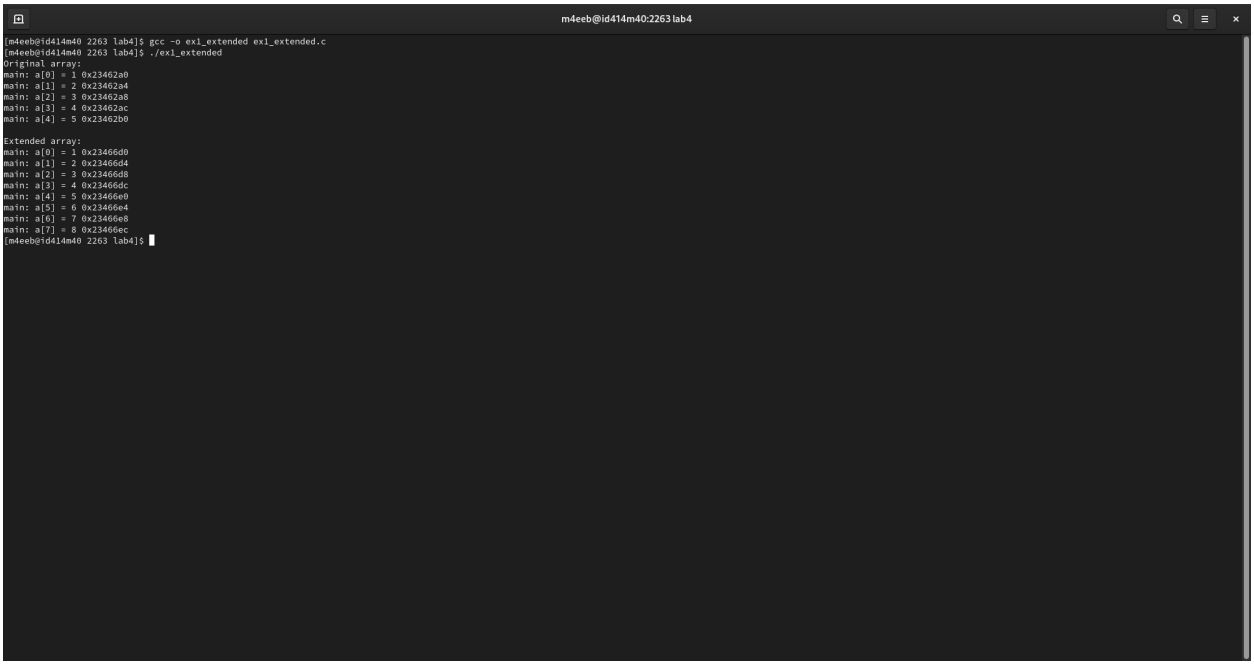


No, the addresses of array elements are not falling within the range of the main function frame. This is because when the array a[ ] is allocated on the heap using malloc(), it is not stored on the stack within the main function's frame. Instead, it is allocated dynamically on the heap, which is separate from the stack frame of the main function. Therefore, the addresses of array elements are not within the range of the main function frame.

Exercise 2

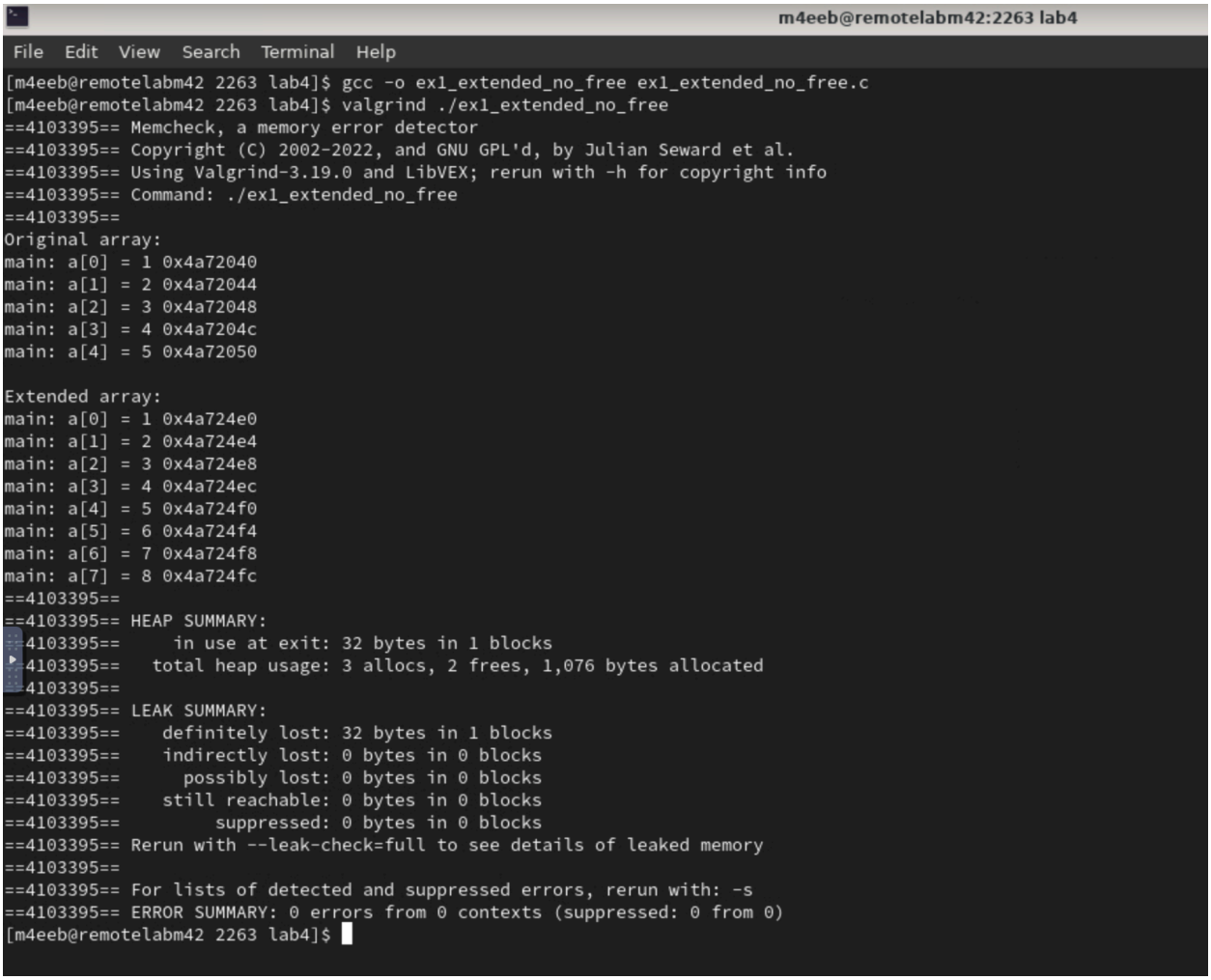
```
#include <stdio.h>
#include <stdlib.h>

void dummy_frame()
{
  return;
}

int main(int argc, char * * argv)
{
  int i;
  int *a = (int *)malloc(5 * sizeof(int));
  if (a == NULL) {
    fprintf(stderr, "Memory allocation failed\n");
    return EXIT_FAILURE;
  }

  printf("Original array:\n");
  for (i = 0; i < 5; i++) {
    a[i] = i + 1;
    printf("main: a[%d] = %d %p \n", i, a[i], &a[i]);
  }

  int *temp = (int *)realloc(a, 8 * sizeof(int));
  if (temp == NULL) {
    fprintf(stderr, "Memory reallocation failed\n");
    free(a);
    return EXIT_FAILURE;
  } else {
    a = temp;
  }

  for (i = 5; i < 8; i++) {
    a[i] = i + 1;
  }

  printf("\nExtended array:\n");
  for (i = 0; i < 8; i++) {
    printf("main: a[%d] = %d %p \n", i, a[i], &a[i]);
  }

  free(a);
  return EXIT_SUCCESS;
```

```
}
```



The addresses for the new extended array may not be the same as the original array. When you extend the array using realloc(), the memory allocator might allocate a new memory block and copy the contents of the original array to the new memory block. In such a case, the addresses of the elements in the extended array would be different from the original array. However, if `realloc()` can extend the original memory block without moving it, the addresses might remain the same. This behavior depends on various factors like the availability of contiguous memory, system memory management strategies, etc.

Exercise 3


m4eeb@remotelabm42:2263 lab4
File Edit View Search Terminal Help
[m4eeb@remotelabm42 2263 lab4]$ gcc -o ex1_extended_no_free ex1_extended_no_free.c
[m4eeb@remotelabm42 2263 lab4]$ valgrind ./ex1_extended_no_free
==4103395== Memcheck, a memory error detector
==4103395== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==4103395== Using Valgrind-3.19.0 and LibVEX; rerun with -h for copyright info
==4103395== Command: ./ex1_extended_no_free
==4103395==
Original array:
main: a[0] = 1 0x4a72040
main: a[1] = 2 0x4a72044
main: a[2] = 3 0x4a72048
main: a[3] = 4 0x4a7204c
main: a[4] = 5 0x4a72050
Extended array:
main: a[0] = 1 0x4a724e0
main: a[1] = 2 0x4a724e4
main: a[2] = 3 0x4a724e8
main: a[3] = 4 0x4a724ec
main: a[4] = 5 0x4a724f0
main: a[5] = 6 0x4a724f4
main: a[6] = 7 0x4a724f8
main: a[7] = 8 0x4a724fc
==4103395==
==4103395== HEAP SUMMARY:
==4103395== in use at exit: 32 bytes in 1 blocks
==4103395== total heap usage: 3 allocs, 2 frees, 1,076 bytes allocated
==4103395==
==4103395== LEAK SUMMARY:
==4103395== definitely lost: 32 bytes in 1 blocks
==4103395== indirectly lost: 0 bytes in 0 blocks
==4103395== possibly lost: 0 bytes in 0 blocks
==4103395== still reachable: 0 bytes in 0 blocks
==4103395== suppressed: 0 bytes in 0 blocks
==4103395== Rerun with --leak-check=full to see details of leaked memory
==4103395==
==4103395== For lists of detected and suppressed errors, rerun with: -s
==4103395== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
[m4eeb@remotelabm42 2263 lab4]$

3.2

#source code

```
#include <stdio.h>
#include <stdlib.h>

void dummy_frame()
{
  return;
}

int main(int argc, char * * argv)
{
  int i;
  int *a = (int *)malloc(5 * sizeof(int));
  if (a == NULL) {
    fprintf(stderr, "Memory allocation failed\n");
    return EXIT_FAILURE;
  }

  printf("Original array:\n");
  for (i = 0; i < 5; i++) {
    a[i] = i + 1;
    printf("main: a[%d] = %d %p \n", i, a[i], &a[i]);
  }

  int *temp = (int *)realloc(a, 8 * sizeof(int));
  if (temp == NULL) {
    fprintf(stderr, "Memory reallocation failed\n");
    free(a);
    return EXIT_FAILURE;
  } else {
    a = temp;
  }

  for (i = 5; i < 8; i++) {
    a[i] = i + 1;
  }

  printf("\nExtended array:\n");
  for (i = 0; i < 8; i++) {
    printf("main: a[%d] = %d %p \n", i, a[i], &a[i]);
  }
```

```
  free(a);

  return EXIT_SUCCESS;
}
```