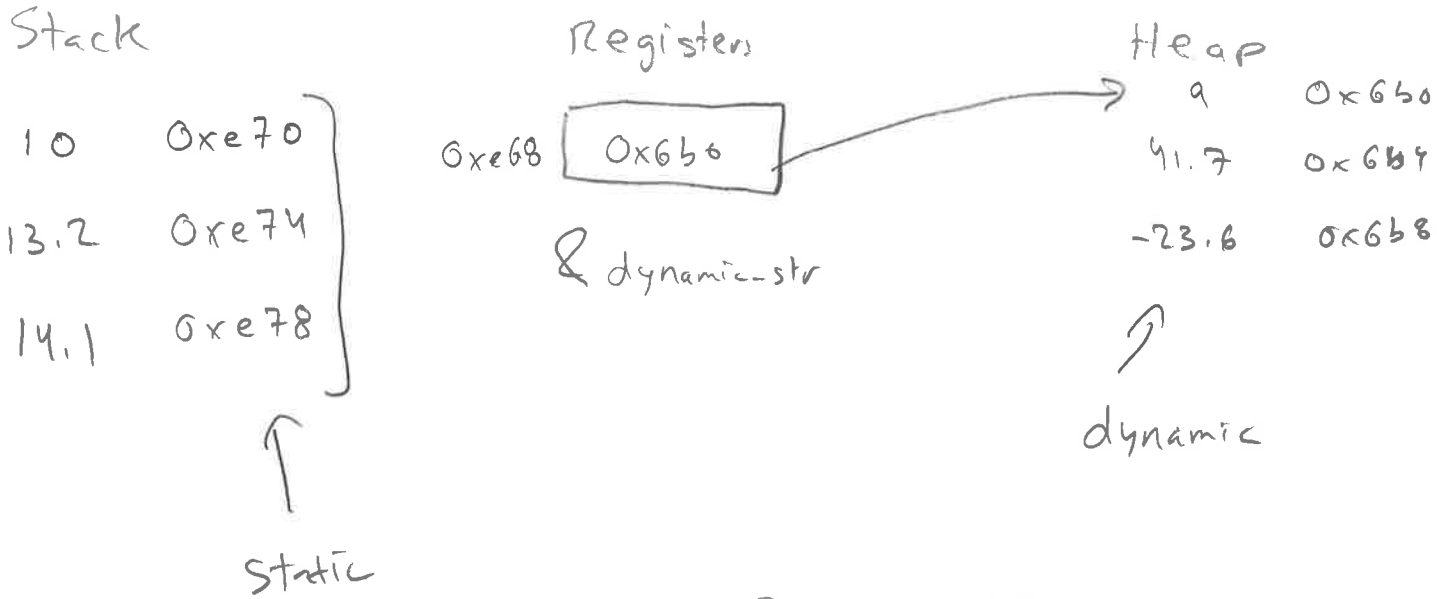
Given the following C code and output run, draw the corresponding memory map

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

typedef struct new_struct{
  int the_int;
  float the_float;
  double the_double;
}new_struct;


int main( const int argc, const char* argv[] ){

  new_struct static_str = {
    atoi(argv[1]),
    (float)atof(argv[2]),
    atof(argv[3])
  };

① fprintf(stdout, "%p\n", &static_str );
② fprintf(stdout, "%p %d\n", &static_str.the_int, static_str.the_int );
③ fprintf(stdout, "%p %f\n", &static_str.the_float, static_str.the_float );
④ fprintf(stdout, "%p %lf\n", &static_str.the_double, static_str.the_double );

      // Step 4: Dynamically Allocate a NEW_STRUCT
  new_struct* dynamic_str = (new_struct *)malloc( sizeof(new_struct) );

      // Step 6: De-reference and set values for the int, long unsigned int, and float
  dynamic_str->the_int = atoi( argv[4] );
  dynamic_str->the_float = (float)atof( argv[5] );
  dynamic_str->the_double = atof( argv[6] );

⑤ fprintf( stdout, "%p %p\n", &dynamic_str, dynamic_str );
⑥ fprintf( stdout, "%p %d\n", &dynamic_str->the_int, dynamic_str->the_int );
⑦ fprintf( stdout, "%p %f\n", &dynamic_str->the_float, dynamic_str->the_float );
⑧ fprintf( stdout, "%p %lf\n", &dynamic_str->the_double, dynamic_str->the_double );

  free( dynamic_str );

  return EXIT_SUCCESS;

};
```

Output: from ./structs 10 13.2 14.1 9 47.7 -23.6

```
① 0x7ffd61bb1e70
② 0x7ffd61bb1e70 10
③ 0x7ffd61bb1e74 13.200000
④ 0x7ffd61bb1e78 14.100000
⑤ 0x7ffd61bb1e68 0x5565663a46b0
⑥ 0x5565663a46b0 9
⑦ 0x5565663a46b4 47.700001
⑧ 0x5565663a46b8 -23.600000
```

Page 1 of 2
- Mapping the print statements to the output statements



- Memory Map Answer