

Brandon London  
Galina project 5  
11/13/2019

### **Code for Project 5:**

```
#include<stdio.h>

#include<stdlib.h>

#define SIZE 1000

void f1()

{

    char arr[SIZE];

    static int n = 1;

    static long int addr;


    printf("Call #%%d\t at %p \n",n,arr);

    printf("AR size #%%d\t is %ld\n",n,addr-(long)(arr));

    n++;

    addr = (long)(arr);

    if(n<11)

        f1();

    else return;

}


void f2()

{

    char arr[SIZE];
```

```
static int n = 1;
```

```
static long int addr;
```

```
printf("Call #%%d\t at %p \n",n,arr);
```

```
printf("AR size #%%d\t is %ld\n",n,addr-(long)(arr));
```

```
printf("Stack Size\t#%d is %ld\n",n,n*addr);
```

```
n++;
```

```
addr = (long)(arr);
```

```
f2();
```

```
}
```

```
void f3()
```

```
{
```

```
char *arr = (char*) malloc(sizeof(char)*SIZE);
```

```
static int n = 1;
```

```
static long int addr;
```

```
printf("Call #%%d\t at %p \n",n,arr);
```

```
printf("AR size #%%d\t is %ld\n",n,addr-(long)(arr));
```

```
n++;
```

```
addr = (long)(arr);
```

```
free(arr);
```

```
if(n<11)
```

```
    f3();
```

```
else
```

```

        return;

}

int main()

{

    f1();

    // f2();

    // f3();

    return 0;

}

```

### Output f1():

```

Call #1  at 0x7ffd0b4de600
AR size #1      is -140724793107968
Call #2  at 0x7ffd0b4de200
AR size #2      is 1024
Call #3  at 0x7ffd0b4dde00
AR size #3      is 1024
Call #4  at 0x7ffd0b4dda00
AR size #4      is 1024
Call #5  at 0x7ffd0b4dd600
AR size #5      is 1024
Call #6  at 0x7ffd0b4dd200
AR size #6      is 1024
Call #7  at 0x7ffd0b4dce00
AR size #7      is 1024
Call #8  at 0x7ffd0b4dca00
AR size #8      is 1024
Call #9  at 0x7ffd0b4dc600
AR size #9      is 1024
Call #10      at 0x7ffd0b4dc200
AR size #10     is 1024
[bcl5zb@delmar 4250]$ 

```

### Output f2():

```

Stack Size      #8168 is 1149516748271967104
Call #8169      at 0x7fff3a8f0430
AR size #8169   is 1024
Stack Size      #8169 is 1149657482439443376
Call #8170      at 0x7fff3a8f0030
AR size #8170   is 1024
Stack Size      #8170 is 1149798216606917600
Call #8171      at 0x7fff3a8efc30
AR size #8171   is 1024
Stack Size      #8171 is 1149938950774389776
Call #8172      at 0x7fff3a8ef830
AR size #8172   is 1024
Stack Size      #8172 is 1150079684941859904
Call #8173      at 0x7fff3a8ef430
AR size #8173   is 1024
Stack Size      #8173 is 1150220419109327984
Call #8174      at 0x7fff3a8ef030
AR size #8174   is 1024
Stack Size      #8174 is 1150361153276794016
Call #8175      at 0x7fff3a8eec30
AR size #8175   is 1024
Stack Size      #8175 is 1150501887444258000
Call #8176      at 0x7fff3a8ee830
AR size #8176   is 1024
Stack Size      #8176 is 1150642621611719936
Call #8177      at 0x7fff3a8ee430
AR size #8177   is 1024
Stack Size      #8177 is 1150783355779179824
Call #8178      at 0x7fff3a8ee030
AR size #8178   is 1024
Stack Size      #8178 is 1150924089946637664
Call #8179      at 0x7fff3a8edc30
AR size #8179   is 1024
Stack Size      #8179 is 1151064824114093456
Call #8180      at 0x7fff3a8ed830
AR size #8180   is 1024
Stack Size      #8180 is 1151205558281547200
Segmentation fault

```

**Output f3():**

```
Call #1 at 0x724010
AR size #1 is -7487504
Call #2 at 0x724010
AR size #2 is 0
Call #3 at 0x724010
AR size #3 is 0
Call #4 at 0x724010
AR size #4 is 0
Call #5 at 0x724010
AR size #5 is 0
Call #6 at 0x724010
AR size #6 is 0
Call #7 at 0x724010
AR size #7 is 0
Call #8 at 0x724010
AR size #8 is 0
Call #9 at 0x724010
AR size #9 is 0
Call #10 at 0x724010
AR size #10 is 0
```