Task 3.

The purpose of this problem is to familiarize with the gdb debugger on Euler and to understand better how pointer arithmetic works. To this end, you will have to use the flag `-g` when compiling your code with `gcc` to include debug information in the executable. Consider the code in the text-box at right. Use the gdb debugger to step through the code and answer the following questions:

a) What is the size of variable p on Euler?
b) What is the address of p?
c) What is the address of c?
d) What is the value of `arr[0]` after the assignment on line 16?
e) What is the value of `arr[0]` at the end of the program?
f) Explain why the value of `arr[0]` changes

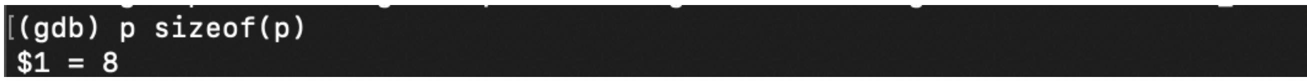
```c
int main() {
  int d;
  char c;
  short s;
  int *p;
  int arr[2];

  p = &d;
  *p = 10;
  c = (char)1;

  p = arr;
  *(p + 1) = 5;
  p[0] = d;

  *((char *)p + 1) = c;

  return 0;
}
```

Report your responses to questions a) through f) in your written response.

Notes:

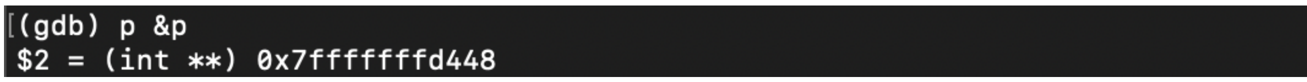
- The source code shown in the image above is available as task3.c in the upstream repository. It can be compiled with the following command:
  ```
  gcc -g task3.c -o task3
  ```
- gdb should be invoked with Slurm using the following command:
  ```
  srun -p slurm_shortgpu --pty -u gdb task3
  ```

a) What is the size of variable p on Euler?
8



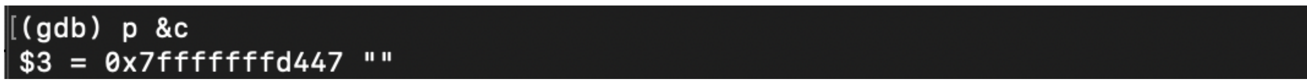
```
[(gdb) p sizeof(p)
 $1 = 8
```

b) What is the address of p?
`0x7fffffffd448`



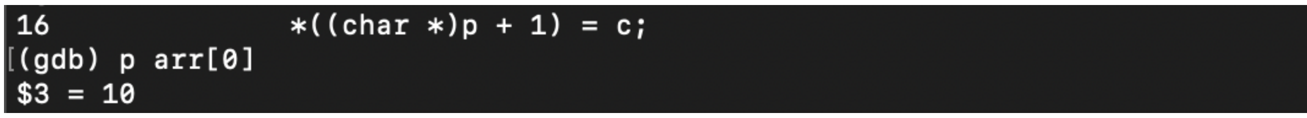
```
[(gdb) p &p
 $2 = (int **) 0x7fffffffd448
```

c) What is the address of c?
`0x7fffffffd447`



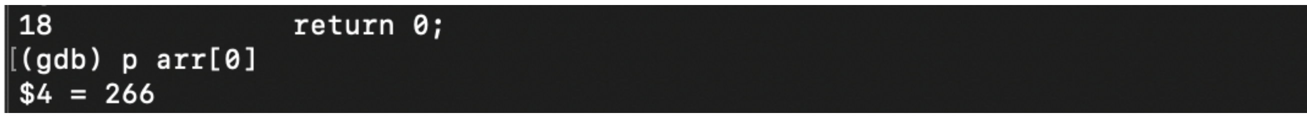
```
[(gdb) p &c
 $3 = 0x7fffffffd447 ""
```

d) What is the value of `arr[0]` after the assignment on line 16?
10



```
 16                  *((char *)p + 1) = c;
[(gdb) p arr[0]
 $3 = 10
```

e) What is the value of `arr[0]` at the end of the program?
266



```
 18                  return 0;
[(gdb) p arr[0]
 $4 = 266
```

f) Explain why the value of `arr[0]` changes
The value of