```c
/**
 * file: radius.c
 *
 * Created by hengxin on 11/24/23.
 */

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

#define PI 3.14

int main() {
  /********** On radius **********/
  int radius = 100;

  printf("radius = %d\n", radius);

  // every variable has an address
  // &: address-of operator ("□□□"□□□)
  printf("&radius = %p\n", &radius);
  // we have already used the address of a variable before
  // scanf("%d", &radius);

  // radius as a left value; refer to its address (the storage space)
  radius = 200;
  // radius as a right value; refer to its value
  double circumference = 2 * PI * radius;
  printf("circumference = %f\n", circumference);
  /********** On radius **********/

  /********** On ptr_radius1 **********/
  // ptr_radius1 is a variable of type "pointer to int"
  int *ptr_radius1 = &radius;
  // ptr_radius1 is a variable: has its value
  printf("ptr_radius1 = %p\n", ptr_radius1);
  // ptr_radius1 is a variable: has its address
  printf("The address of ptr_radius1 is %p\n", &ptr_radius1);
  /********** On ptr_radius1 **********/

  /********** On *ptr_radius1 **********/
  // IMPORTANT:
  // *ptr_radius1: behaves just like radius
  // type: int; value: the value of radius; address: the address of
radius
  // *: indirection/dereference operator ("□□□□"/"□□□"□□□)
  printf("radius = %d\n", *ptr_radius1);
  // *ptr_radius1 as a right value
  circumference = 2 * 3.14 * (*ptr_radius1);
  // take the address of *ptr_radius1
  // &*ptr_radius1 is the same as ptr_radius1
  printf("The address of *ptr_radius1 is %p\n", &*ptr_radius1);
  // *ptr_radius1 as a left value
  *ptr_radius1 = 100;
```

```c
    printf("radius = %d\n", *ptr_radius1);
    /********** On *ptr_radius1 **********/

    /********** Begin: On ptr_radius1 as lvalue and rvalue **********/
    // ptr_radius1 as a left value
    int radius2 = 200;
    int *ptr_radius2 = &radius2;

    ptr_radius1 = ptr_radius2;
    printf("radius = %d\n", *ptr_radius1);

    // ptr_radius1 as a right value
    ptr_radius2 = ptr_radius1;
    printf("radius = %d\n", *ptr_radius2);
    /********** Begin: On ptr_radius1 as lvalue and rvalue **********/

    /********** On array names **********/
    int numbers[5] = {0};
    // vs. numbers[2] = {2};
    // numbers++;
    // numbers = &radius;
    int *ptr_array = numbers;
    ptr_array++;
    /********** On array names **********/

    /********** On malloc/free **********/
    // undefined behavior
    // free(numbers);
    /********** On malloc/free **********/

    /********** On const **********/
    // const int * and int const *
    // You cannot modify the value pointed to by ptr_radius3
    // through the pointer (without casting the constness away).
    const int *ptr_radius3 = &radius;
    // *ptr_radius is read-only
    // *ptr_radius3 = 300;
    // You are allowed to do this, but you should not do it!
    int *ptr_radius4 = ptr_radius3;
    *ptr_radius4 = 400;
    printf("radius = %d\n", radius);

    // int * const
    int *const ptr_radius5 = &radius;
    // ptr_radius5 = ptr_radius3;
    *ptr_radius5 = 500;
    printf("radius = %d\n", radius);

    // const int * const
    const int *const ptr_radius6 = &radius;
    // ptr_radius6 = ptr_radius3;
    // *ptr_radius6 = 600;
    /********** On const **********/
```

```
106
107   return 0;
108 }
```