```
runner% cat quad.c
/* quad: tell the quadrant an input pair */
#include <stdio.h>

void get_point(double *xp, double *yp);
int quad1(double x, double y);
int quad2(double x, double y);
int quad3(double x, double y);
int quad4(double x, double y);
void print_result(int method, int q);

int main(void)
{
   double x, y;
   int q;
   get_point(&x, &y);
   q = quad1(x, y); print_result(1, q);
   q = quad2(x, y); print_result(2, q);
   q = quad3(x, y); print_result(3, q);
   q = quad4(x, y); print_result(4, q);
   return 0;
}

/* get_point: read and return point */
void get_point(double *xp, double *yp)
{
   scanf("%lf %lf", xp, yp);
}

/* quad1: determine quadrant */
int quad1(double x, double y)
{
   int r;
   if (x == 0 || y == 0) r = 0;
   if (x > 0 && y > 0) r = 1;
   if (x > 0 && y < 0) r = 4;
   if (x < 0 && y > 0) r = 2;
   if (x < 0 && y < 0) r = 3;
   return r;
}

/* quad2: determine quadrant */
int quad2(double x, double y)
{
   int r;
   if (x == 0 || y == 0) r = 0;
   else if (x > 0) {
      if (y > 0) r = 1;
      else if (y < 0) r = 4;
   }
   else if (x < 0) {
      if (y > 0) r = 2;
      else if (y < 0) r = 3;
   }
   return r;
}

/* quad3: determine quadrant */
int quad3(double x, double y)
{
   int r;
   if (x == 0 || y == 0) r = 0;
   else if (x > 0) {
      if (y > 0) r = 1;
      else r = 4;
   }
   else {
      if (y > 0) r = 2;
      else r = 3;
   }
   return r;
}

/* quad4: determine quadrant */
int quad4(double x, double y)
{
   if (x == 0 || y == 0) return 0;
   if (x > 0) {
      if (y > 0) return 1;
      return 4;
   }
   if (y > 0) return 2;
   return 3;
}

/* print_result: method # and quad # */
void print_result(int method, int q)
{
   printf("Method %d: ", method);
   if (q == 0) printf("On and axis\n");
   else printf("Quadrant %d\n", q);
}
runner% cc -o quad quad.c
```

```
runner% quad
1  2
Method 1: Quadrant 1
Method 2: Quadrant 1
Method 3: Quadrant 1
Method 4: Quadrant 1
runner% quad
-1 3
Method 1: Quadrant 2
Method 2: Quadrant 2
Method 3: Quadrant 2
Method 4: Quadrant 2
runner% quad
3 -2
Method 1: Quadrant 4
Method 2: Quadrant 4
Method 3: Quadrant 4
Method 4: Quadrant 4
runner% quad
-1 -4
Method 1: Quadrant 3
Method 2: Quadrant 3
Method 3: Quadrant 3
Method 4: Quadrant 3
---------------------------------------------------
runner% cat sphere0.c
#include <stdio.h>
#include <stdlib.h>
#define PI 3.1415926535
int main()
{
        double r, v;
        printf("Enter radius of a sphere>>>");
        scanf("%lf", &r);
        if (r <= 0.0) {
           printf("Radius must be > zero\n");
           exit(1);
        }
        v = (4.0/3.0)*PI*r*r*r;
        printf("Volume: %.4f\n", v);
        return 0;
}
----------------------------------------------------
runner% cc -o sphere0 sphere0.c
runner% sphere0
Enter radius of a sphere>>>1.0
Volume: 4.1888
runner% sphere0
Enter radius of a sphere>>>10.0
Volume: 4188.7902
```

```
runner% cat sphere.c
/* Produces exactly the same output as sphere0.c */
#include <stdio.h>
#include <stdlib.h>
#define PI 3.1415926535

double get_radius(void);
double calc_vol(double);
void print_vol(double);
void radius_error(void);

int main()
{
        double r, v;
        r = get_radius();
        if (r <= 0.0) radius_error();
        v = calc_vol(r);
        print_vol(v);
        return 0;
}

double get_radius(void)
{
        double r;
        printf("Enter radius of a sphere>>>");
        scanf("%lf", &r);
        return r;
}

double calc_vol(double r)
{
        double v;
        v = (4.0/3.0)*PI*r*r*r;
        return v;
}

void print_vol(double v)
{
        printf("Volume: %.4f\n", v);
}

void radius_error(void)
{
        printf("Radius must be > zero\n");
        exit(1);
}
```