ex1-1.c

```c
/* code: ex1-1.c    (v1.20.00) */
#include <stdio.h>

int main ()
{
  printf ("The Open University of Japan\n");

  return 0;
}
```

ex1-2.c

```
/* code: ex1-2.c   (v1.20.00) */
#include <stdio.h>

int main ()
{
  int a;
  printf ("Enter an integer: ");
  scanf ("%d", &a);
  printf ("The integer you entered was %d.\n", a);

  return 0;
}

/* In Visual Studio (C++), it is recommended to
use the scanf_s() function instead of the scanf()
function used in Chapter 1.
However, it is possible to use scanf() with
the setting #pragma warning(disable:4996). */
```

ex1-3.c

```c
/* code: ex1-3.c    (v1.20.00) */
#include <stdio.h>

int main ()
{
  char a;
  short b;
  int c;
  long d;
  float e;
  double f;
  printf ("char:   %zd byte(s)\n", sizeof (a));
  printf ("short:  %zd byte(s)\n", sizeof (b));
  printf ("int:    %zd byte(s)\n", sizeof (c));
  printf ("long:   %zd byte(s)\n", sizeof (d));
  printf ("float:  %zd byte(s)\n", sizeof (e));
  printf ("double: %zd byte(s)\n", sizeof (f));

  return 0;

}
```

ex1-4.c

```
/* code: ex1-4.c    (v1.20.00) */
#include <stdio.h>

int main ()
{
  int a, b, c;
  a = 10;
  b = 3;
  c = 0;
  printf ("a=%d\n", a);
  printf ("b=%d\n\n", b);
  c = a + b;
  printf ("a + b = %d\n", c);
  c = a - b;
  printf ("a - b = %d\n", c);
  c = a * b;
  printf ("a * b = %d\n", c);
  c = a / b;
  printf ("a / b = %d\n", c);
  c = a % b;
  printf ("a %% b = %d\n", c);

  return 0;
}
```

ex1-5.c

```c
/* code: ex1-5.c    (v1.20.00) */
#include <stdio.h>
#include <math.h>

int main ()
{
  double x, y, z;
  x = 30.0;
  y = 3.0;
  z = 0.0;
  printf ("x=%f\n", x);
  printf ("y=%f\n\n", y);
  z = pow (x, y);
  printf ("pow(x,y) = %f\n", z);

  return 0;
}
```

ex1-6.c

```
/* code: ex1-6.c    (v1.20.00) */
#include <stdio.h>
#include <math.h>

int main ()
{
  float celsius, fahrenheit;

  celsius = 36.5;
  fahrenheit = (9.0 / 5.0) * celsius + 32.0;
  printf ("%f(Celsius) = %f(Fahrenheit)\n", celsius,
     fahrenheit);

  return 0;
}
```

ex1-7.c

```c
/* code: ex1-7.c    (v1.20.00) */
#include <stdio.h>

int main ()
{
  printf ("The Open University of Japan\n");
  /* web address
      http://www.ouj.ac.jp/   */

  // C++ style comments
  // C99 allows single-line comments

  return 0;
}
```

q1-1.c

```
/* code: q1-1.c    (v1.20.00) */
#include <stdio.h>
#include <math.h>

int main ()
{
  double x, y;

  x = 3.14159;
  y = 0.0;
  printf ("x=%f\n\n", x);
  y = ceil (x);
  printf ("ceil(x)  = %f\n", y);
  y = floor (x);
  printf ("floor(x) = %f\n", y);

  return 0;
}
```

q1-2.c

```c
/* code: q1-2.c    (v1.20.00) */
#include <stdio.h>
#include <math.h>

int main ()
{
  float fx, fz;
  double dx, dz;
  long double lx, lz;

  fx = 100.00F;
  fz = sqrtf (fx);
  printf ("fx = %f\n", fx);
  printf ("sqrtf(fx) = %f\n\n", fz);

  dx = 100.00;
  dz = sqrt (dx);
  printf ("dx = %f\n", dx);
  printf ("sqrt(dx)  = %f\n\n", dz);

  lx = 100.00L;
  lz = sqrtl (lx);
  printf ("lx = %Lf\n", lx);
  printf ("sqrtl(lx) = %Lf\n\n", lz);

  return 0;
}
```

q1-3.c

```c
/* code: q1-3.c   (v1.20.00) */
#include <stdio.h>
#include <math.h>

int main ()
{
  float fahrenheit, celsius;

  fahrenheit = 25.1;
  celsius = (5.0 / 9.0) * (fahrenheit - 32.0);
  printf ("%f(Fahrenheit) = %f(Celsius)\n", fahrenheit,
      celsius);

  return 0;
}
```

ex2-1.c

```c
/* code: ex2-1.c   (v1.20.00) */
#include <stdio.h>

int main ()
{
  int x, y;

  x = 500;
  y = 300;

  printf ("X = %d\n", x);
  printf ("Y = %d\n", y);

  if (x > y)
    printf ("X is greater than Y.\n");


  return 0;
}
```

ex2-2.c

```c
/* code: ex2-2.c   (v1.20.00) */
#include <stdio.h>

int main ()
{
  int x, y;
  x = 500;
  y = 700;
  printf ("X = %d\n", x);
  printf ("Y = %d\n", y);

  if (x > y)
    printf ("X is greater than Y.\n");
  else
    printf ("X is less than or equal to Y.\n");

  return 0;
}
```

ex2-3.c

```
/* code: ex2-3.c   (v1.20.00) */
#include <stdio.h>

int main ()
{
  char grade;

  grade = 'B';

  switch (grade) {
  case 'A':
    printf ("excellent\n");
    break;
  case 'B':
    printf ("good\n");
    break;
  case 'C':
    printf ("fair\n");
    break;
  case 'D':
    printf ("barely passing\n");
    break;
  case 'F':
    printf ("not passing\n");
    break;
  default:
    printf ("ERROR: invalid character\n");
    break;
  }
  printf ("Your grade is  %c\n", grade);
  return 0;
}
```

q2-1.c

```
/* code: q2-1.c   (v1.20.00) */
#include <stdio.h>

int main ()
{
  int x, y;

  printf ("enter X: ");
  scanf ("%d", &x);
  printf ("enter Y: ");
  scanf ("%d", &y);

  printf ("X = %d\n", x);
  printf ("Y = %d\n", y);

  if (x > y) {
    printf ("X is greater than Y.\n");
  }
  else {
    printf ("X is less than or equal to Y.\n");
  }

  return 0;
}
```

q2-2.c

```
/* code: q2-2.c    (v1.20.00) */
#include <stdio.h>

int main ()
{
  char grade;

  grade = 'b';

  switch (grade) {
  case 'a':
  case 'A':
    printf ("excellent\n");
    break;
  case 'b':
  case 'B':
    printf ("good\n");
    break;
  case 'c':
  case 'C':
    printf ("fair\n");
    break;
  case 'd':
  case 'D':
    printf ("barely passing\n");
    break;
  case 'f':
  case 'F':
    printf ("not passing\n");
    break;
  default:
    printf ("ERROR: invalid character\n");
    break;
  }
  printf ("Your grade is  %c\n", grade);
```

```
    return 0;
}
```

q2-3.c

```c
/* code: q2-3.c    (v1.20.00) */
#include <stdio.h>

int main ()
{
  int a;
  a = 3;

  if (a == 0 || a == 1 || a == 2) {
    printf ("A\n");
  }
  else if (a == 3 || a == 4) {
    printf ("B\n");
  }
  else {
    printf ("ERROR: invalid number\n");
  }

  return 0;
}
```

q2-3a.c

```c
/* code: q2-3a.c   (v1.20.00) */
#include <stdio.h>

int main ()
{
  int a;
  a = 3;
  switch (a) {
  case 0:
  case 1:
  case 2:
    printf ("A\n");
    break;
  case 3:
  case 4:
    printf ("B\n");
    break;
  default:
    printf ("ERROR: invalid number\n");
    break;
  }

  return 0;
}
```

ex3-1.c

```c
/* code: ex3-1.c   (v1.20.00) */
#include <stdio.h>

int main ()
{
  int i;

  for (i = 0; i < 10; i++)
    printf ("%d ", i);

  return 0;
}
```

ex3-10.c

```c
/* code: ex3-10.c    (v1.20.00) */
#include <stdio.h>

int main ()
{
  int i;

  i = 0;
  while (1) {
    printf ("%d ", i);
    if (i == 5) {
      i = 0;
      break;
    }
    i++;
  }

  return 0;
}
```

ex3-11.c

```c
/* code: ex3-11.c    (v1.20.00) */
#include <stdio.h>

int main ()
{
  int i;

  i = 0;
  while (1) {
    printf ("%d-", i);
    if (i == 5) {
      i = 0;
      continue;
    }
    i++;
  }

  return 0;
}
```

ex3-2.c

```
/* code: ex3-2.c   (v1.20.00) */
#include <stdio.h>

int main ()
{
  int i;

  for (i = 0; i < 10; i++) {
    printf ("%d", i);
    if (0 != (i % 2))
      printf (":odd ");
    else
      printf (":even ");
  }

  return 0;
}
```

ex3-3.c

```c
/* code: ex3-3.c   (v1.20.00) */
#include <stdio.h>

int main ()
{
  int i, j;

  for (i = 1; i < 10; i++) {
    for (j = 1; j < 10; j++) {
      printf ("%02d ", i * j);
    }
    printf ("\n");
  }

  return 0;
}
```

ex3-4.c

```
/* code: ex3-4.c    (v1.20.00) */
#include <stdio.h>

int main ()
{
  int i;

  i = 0;
  for (;;) {
    printf ("%d ", i);
    i++;
  }
  return 0;
}
```

ex3-5.c

```c
/* code: ex3-5.c    (v1.20.00) */
#include <stdio.h>

int main ()
{
  int i;

  i = 0;
  while (i < 10)
    printf ("%d␣", i++);

  return 0;
}
```

ex3-6.c

```
/* code: ex3-6.c   (v1.20.00) */
#include <stdio.h>

int main ()
{
  int i;

  i = 0;
  while (i < 10) {
    printf ("%d ", i);
    i++;
  }
  return 0;
}
```

ex3-7.c

```c
/* code: ex3-7.c   (v1.20.00) */
#include <stdio.h>

int main ()
{
  int i;

  i = 0;
  do {
    printf ("%d ", i);
    i++;
  } while (i < 10);

  return 0;
}
```

ex3-8.c

```c
/* code: ex3-8.c    (v1.20.00) */
#include <stdio.h>

int main ()
{
  int i, j;

  i = 1;
  while (i < 10) {
    j = 1;
    while (j < 10) {
      printf ("%02d ", i * j);
      j++;
    }
    printf ("\n");
    i++;
  }

  return 0;
}
```

ex3-9.c

```c
/* code: ex3-9.c    (v1.20.00) */
#include <stdio.h>

int main ()
{
  int i;

  i = 0;
  while (1) {
    printf ("%d ", i);
    i++;
  }

  return 0;
}
```

q3-1.c

```c
/* code: q3-1.c    (v1.20.00) */
#include <stdio.h>

int main ()
{
  int i;

  for (i = 0; i < 100; i++)
    printf ("%d ", i);

  return 0;
}
```

q3-2.c

```c
/* code: q3-2.c    (v1.20.00) */
#include <stdio.h>

int main ()
{
  int i;

  for (i = 9; i >= 0; i--)
    printf ("%d ", i);

  return 0;
}
```

q3-3.c

```
/* code: q3-3.c    (v1.20.00) */
#include <stdio.h>

int main ()
{
  int i;

  i = 0;
  while (i < 100) {
    printf ("%d ", i);
    i++;
  }

  return 0;
}
```

q3-4.c

```c
/* code: q3-4.c    (v1.20.00) */
#include <stdio.h>

int main ()
{
  int i, j, k;

  for (i = 0; i < 2; i++) {
    for (j = 0; j < 2; j++) {
      for (k = 0; k < 2; k++) {
        printf ("%d-%d-%d", i, j, k);
        printf ("\n");
      }
    }
  }

  return 0;
}
```

q3-5.c

```
/* code: q3-5.c    (v1.20.00) */
#include <stdio.h>

int main ()
{
  int i, j, k;

  for (i = 0; i < 2; i++) {
    for (j = 0; j < 2; j++) {
      for (k = 0; k < 2; k++) {
        printf ("%d␣", i * j + k);
      }
    }
  }

  return 0;
}
```

ex4-1.c

```
/* code: ex4-1.c   (v1.20.00) */
#include <stdio.h>
#include <stdlib.h>

#define POINTS 1000

int main ()
{
  int i, count, points;
  double x, y, q;
  double pi;

  points = POINTS;
  count = 0;

  for (i = 0; i < points; i++) {
    x = (double) rand () / ((double) RAND_MAX + 1.0);
    y = (double) rand () / ((double) RAND_MAX + 1.0);
    q = (x * x) + (y * y);

    if (q <= 1.00)
      count++;
  }

  pi = (double) count / (double) points *(double) 4.00;
  printf ("circle: %d\t", count);
  printf ("square: %d\t", points);
  printf ("PI: %f\n", pi);

  return 0;
}
```

ex4-2.c

```
/* code: ex4-2.c    (v1.20.00) */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main ()
{
  int i, j, count, points;
  double x, y, q;
  double pi;

  for (j = 1; j < 10; j++) {
    points = 1;
    count = 0;
    points = points * pow (10, j);
    for (i = 0; i < points; i++) {
      x = (double) rand () / ((double) RAND_MAX + 1.0);
      y = (double) rand () / ((double) RAND_MAX + 1.0);
      q = (x * x) + (y * y);

      if (q <= 1.00)
        count++;
    }
    pi = (double) count / (double) points *(double)
       4.00;
    printf ("circle: %10d\t", count);
    printf ("square: %10d\t", points);
    printf ("PI: %f (%+f)\n", pi, (pi - M_PI));
  }
  return 0;
}
```

ex5-1.c

```
/* code: ex5-1.c   (v1.20.00) */
#include <stdio.h>

int main ()
{
  int i;

  for (i = 0; i < 10; i++)
    printf ("%d ", i);
  printf ("\n");

  for (i = 0; i < 10; i++)
    printf ("%d ", i);
  printf ("\n");

  for (i = 0; i < 10; i++)
    printf ("%d ", i);
  printf ("\n");

  return 0;
}
```

ex5-2.c

```c
/* code: ex5-2.c   (v1.20.00) */
#include <stdio.h>

void print_numbers (void)
{
  int i;

  for (i = 0; i < 10; i++)
    printf ("%d ", i);
  printf ("\n");
}

int main ()
{
  print_numbers ();
  print_numbers ();
  print_numbers ();
  return 0;
}
```

ex5-3.c

```c
/* code: ex5-3.c   (v1.20.00) */
#include <stdio.h>

void print_numbers (void);

int main ()
{
  print_numbers ();
  print_numbers ();
  print_numbers ();
  return 0;
}

void print_numbers (void)
{
  int i;

  for (i = 0; i < 10; i++)
    printf ("%d ", i);
  printf ("\n");
}
```

ex5-4.c

```c
/* code: ex5-4.c    (v1.20.00) */
#include <stdio.h>

void g (void)
{
  int i;
  for (i = 0; i < 3; i++) {
    printf ("a");
  }
}

void f (void)
{
  int i;
  for (i = 0; i < 5; i++) {
    g ();
  }
}

int main (int argc, char **argv)
{
  f ();
  return 0;
}
```

ex5-5.c

```c
/* code: ex5-5.c   (v1.20.00) */
#include <stdio.h>

float triangle (float base, float height)
{
  float c;
  c = (base * height) / 2.000F;
  return c;
}

int main ()
{
  float t;
  t = triangle (3.00, 4.00);
  printf ("triangle = %f\n", t);
  t = triangle (5.00, 6.00);
  printf ("triangle = %f\n", t);

  return 0;
}
```

ex5-6.c

```
/* code: ex5-6.c    (v1.20.00) */
#include <stdio.h>

void add_pass_by_value (int i)
{
  i = i + 1;
}

void add_pass_by_reference (int *i)
{
  *i = *i + 1;
}

int main ()
{
  int a;

  a = 10;
  add_pass_by_value (a);
  printf ("%d\n", a);

  a = 10;
  add_pass_by_reference (&a);
  printf ("%d\n", a);

  return 0;
}
```

ex5-7.c

```c
/* code: ex5-7.c    (v1.20.00) */
#include <stdio.h>

int factorial (int n)
{
  if (n == 0) {
    return 1;
  }
  else {
    return n * factorial (n - 1);
  }
}

int main ()
{
  int i;
  i = 5;
  printf ("%d! = %d\n", i, factorial (i));

  return 0;
}
```

q5-1.c

```c
/* code: q5-1.c    (v1.20.00) */
#include <stdio.h>

float trapezoid (float a, float b, float h)
{
  float c;
  c = ((a + b) / 2.000F) * h;
  return c;
}

int main ()
{
  float t;
  t = trapezoid (3.00, 4.00, 5.00);
  printf ("trapezoid = %f\n", t);
  t = trapezoid (5.00, 6.00, 7.00);
  printf ("trapezoid = %f\n", t);

  return 0;
}
```

q5-2.c

```c
/* code: q5-2.c   (v1.20.00) */
#include <stdio.h>

struct student
{
  int id;
  char grade;
  float average;
};
typedef struct student STUDENT_TYPE;

STUDENT_TYPE initialize_student_record (STUDENT_TYPE s)
{
  s.id++;
  s.grade = 'x';
  s.average = 0.0;
  return s;
}

int main ()
{
  STUDENT_TYPE student;

  student.id = 20;
  student.grade = 'a';
  student.average = 300.000;
  printf ("%d-%c-%f\n", student.id, student.grade,
      student.average);
  student = initialize_student_record (student);
  printf ("%d-%c-%f\n", student.id, student.grade,
      student.average);

  return 0;
}
```

q5-3.c

```
/* code: q5-3.c   (v1.20.00) */
#include <stdio.h>

int fibonacci (int n)
{
  if (n == 0) {
    return 0;
  }
  else if (n == 1) {
    return 1;
  }
  else {
    return (fibonacci (n - 1) + fibonacci (n - 2));
  }
}

int main ()
{
  int i;
  i = 10;
  printf ("fibonacci(%d) = %d\n", i, fibonacci (i));

  return 0;
}
```

q5-4.c

```
/* code: q5-4.c    (v1.20.00) */
#include <stdio.h>

void foo (int n)
{
   if (n < 15) {
      foo (n + 1);
      printf ("%d␣", n);
   }
}

int main ()
{

   foo (0);

   return 0;
}
```

q5-5.c

```
/* code: q5-5.c    (v1.20.00) */
#include <stdio.h>

int factorial (int n)
{
  if (n == 0) {
    return 1;
  }
  else {
    return n * factorial (n - 1);
  }
}

int main ()
{
  int i;
  /* i = -1; */
  i = 1;
  printf ("%d! = %d\n", i, factorial (i));

  return 0;
}
```

ex6-1.c

```c
/* code: ex6-1.c    (v1.20.00) */
#include <stdio.h>

int main ()
{
  int a, b, c, d, e;
  int sum, avg;

  a = 30;
  b = 20;
  c = 10;
  d = 25;
  e = 15;
  sum = a + b + c + d + e;
  avg = sum / 5;
  printf ("%d\n", avg);

  return 0;
}
```

ex6-10.c

```c
/* code: ex6-10.c    (v1.20.00) */
#include <stdio.h>
#include <string.h>

int main ()
{
  char s0[] = "aaaaa";
  char s1[] = "bbbbb";
  char s2[] = "aaaaaaa";
  int i;
  printf ("strcmp(str1, str2)\n");
  i = strcmp (s0, s0);
  printf ("[%s] [%s] (%d)\n", s0, s0, i);
  i = strcmp (s0, s1);
  printf ("[%s] [%s] (%d)\n", s0, s1, i);
  i = strcmp (s1, s0);
  printf ("[%s] [%s] (%d)\n", s1, s0, i);
  i = strcmp (s0, s2);
  printf ("[%s] [%s] (%d)\n", s0, s2, i);

  return 0;
}
```

ex6-2.c

```c
/* code: ex6-2.c    (v1.20.00) */
#include <stdio.h>

int main ()
{
  int a[10];
  int i, sum, avg;

  a[0] = 30;
  a[1] = 20;
  a[2] = 10;
  a[3] = 25;
  a[4] = 15;
  sum = 0;
  for (i = 0; i < 5; i++)
    sum += a[i];

  avg = sum / 5;
  printf ("%d\n", avg);

  return 0;
}
```

ex6-3.c

```c
/* code: ex6-3.c    (v1.20.00) */
#include <stdio.h>

int main ()
{
  int a[10] = { 30, 20, 10, 25, 15 };
  int i, sum, avg;

  sum = 0;
  for (i = 0; i < 5; i++)
    sum += a[i];

  avg = sum / 5;
  printf ("%d\n", avg);

  return 0;
}
```

ex6-4.c

```c
/* code: ex6-4.c    (v1.20.00) */
#include <stdio.h>
#include <stdlib.h>

#define ARRAY_SIZE 10

int main ()
{
  int a[ARRAY_SIZE];
  int i;

  for (i = 0; i < ARRAY_SIZE; i++)
    a[i] = rand () % 100;

  for (i = 0; i < ARRAY_SIZE; i++)
    printf ("%03d ", a[i]);

  return 0;
}
```

ex6-5.c

```c
/* code: ex6-5.c   (v1.20.00) */
#include <stdio.h>

int main ()
{
  int i, j;
  int a[3][4] = {
    {0, 10, 20, 30},
    {40, 50, 60, 70},
    {80, 90, 100, 110}
  };

  for (i = 0; i < 3; i++) {
    for (j = 0; j < 4; j++) {
      printf ("array[%d][%d]=%3d\n", i, j, a[i][j]);
    }
  }

  return 0;
}
```

ex6-6.c

```c
/* code: ex6-6.c   (v1.20.00) */
#include <stdio.h>

int main ()
{
  int i, j, k;
  int a[2][3][4] = {
    {{0, 1, 2, 3},
     {4, 5, 6, 7},
     {8, 9, 10, 11}},
    {{0, 10, 20, 30},
     {40, 50, 60, 70},
     {80, 90, 100, 110}}
  };

  for (i = 0; i < 2; i++) {
    for (j = 0; j < 3; j++) {
      for (k = 0; k < 4; k++) {
        printf ("array[%d][%d][%d]=%3d\n", i, j, k, a[i
            ][j][k]);
      }
    }
  }

  return 0;
}
```

ex6-7.c

```
/* code: ex6-7.c    (v1.20.00) */
#include <stdio.h>

int main ()
{
  char s[4];
  s[0] = 'O';
  s[1] = 'U';
  s[2] = 'J';
  s[3] = '\0';
  printf ("%s\n", s);

  return 0;
}
```

ex6-8.c

```c
/* code: ex6-8.c    (v1.20.00) */
#include <stdio.h>

/* ------------------------------------------------ */
void string_copy (char *target, char *source)
{
  int i;
  i = 0;
  while (source[i] != '\0') {
    target[i] = source[i];
    i++;
  }
  target[i] = '\0';
}

/* ------------------------------------------------ */
int main ()
{
  char s[20] = "University";
  char t[20];

  string_copy (t, s);
  printf ("%s\n", t);

  return 0;
}
```

ex6-9.c

```c
/* code: ex6-9.c    (v1.20.00) */
#include <stdio.h>
#include <string.h>

/* ------------------------------------------------------------ */
int main ()
{
  char s[20] = "University";
  char t[20];
  strcpy (t, s);
  printf ("%s\n", t);

  return 0;
}
```

q6-1.c

```c
/* code: q6-1.c    (v1.20.00) */
#include <stdio.h>

int main ()
{
  float a[5];
  int i;
  float sum, avg;

  a[0] = 30.0;
  a[1] = 20.0;
  a[2] = 10.0;
  a[3] = 25.0;
  a[4] = 15.0;
  sum = 0.0;
  for (i = 0; i < 5; i++)
    sum += a[i];

  avg = sum / 5.00;
  printf ("%f\n", avg);

  return 0;
}
```

q6-2.c

```
/* code: q6-2.c   (v1.20.00) */
#include <stdio.h>
#define TABLE 9
int main ()
{
  int i, j;
  int a[TABLE][TABLE];

  for (i = 0; i < TABLE; i++) {
    for (j = 0; j < TABLE; j++) {
      a[i][j] = (i + 1) * (j + 1);
    }
  }
  for (i = 0; i < TABLE; i++) {
    for (j = 0; j < TABLE; j++) {
      printf ("%02d ", a[i][j]);
    }
    printf ("\n");
  }
  return 0;
}
```

q6-3.c

```c
/* code: q6-3.c    (v1.20.00) */
#include <stdio.h>
#include <stdlib.h>

int main ()
{
  int i, j, k;
  int array[2][3][4];

  for (i = 0; i < 2; i++) {
    for (j = 0; j < 3; j++) {
      for (k = 0; k < 4; k++) {
        array[i][j][k] = (rand () % 100) + 1;
      }
    }
  }
  for (i = 0; i < 2; i++) {
    for (j = 0; j < 3; j++) {
      for (k = 0; k < 4; k++) {
        printf ("%03d ", array[i][j][k]);
      }
      printf ("\n");
    }
    printf ("\n");
  }
  return 0;
}
```

q6-4.c

```
/* code: q6-4.c    (v1.20.00) */
#include <stdio.h>
#include <string.h>

/* ------------------------------------------------------------ */
int main ()
{
  char s0[] = "aaaaa";
  char s1[] = "bbbbb";
  char s2[] = "aaaaaaa";
  int i;
  printf ("strncmp(str1, str2, 3)\n");
  i = strncmp (s0, s0, 3);
  printf ("[%s] [%s] (%d)\n", s0, s0, i);
  i = strncmp (s0, s1, 3);
  printf ("[%s] [%s] (%d)\n", s0, s1, i);
  i = strncmp (s1, s0, 3);
  printf ("[%s] [%s] (%d)\n", s1, s0, i);
  i = strncmp (s0, s2, 3);
  printf ("[%s] [%s] (%d)\n", s0, s2, i);

  return 0;
}
```

q6-5.c

```c
/* code: q6-5.c    (v1.20.00) */
#include <stdio.h>
#include <string.h>

int main ()
{
  char s0[] = "abcdefg";
  int i;
  i = strlen (s0);
  printf ("[%s] (%d)\n", s0, i);
  return 0;
}
```

q6-6.c

```c
/* code: q6-6.c   (v1.20.00) */

/* In Visual Studio (C++), the error (E0513, C2440)
in q6-6.c can be avoided by casting.
Using the strcpy_s() function is also recommended. */

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

#define MAX 10

struct student
{
  int id;
  char grade;
  char name[128];
};
typedef struct student STUDENT_TYPE;

/* ------------------------------------------------ */
int main ()
{
  STUDENT_TYPE db1[MAX];
  STUDENT_TYPE *db2[MAX];
  int i;

  printf ("database1\n");
  for (i = 0; i < MAX; i++) {
    db1[i].id = 100 + i;
    db1[i].grade = 'a' + rand () % 5;
    strcpy (db1[i].name, "John Doe");
    printf ("%d %c %s\n", db1[i].id, db1[i].grade, db1[
        i].name);
  }

  printf ("\n");
  printf ("database2\n");
  for (i = 0; i < MAX; i++) {
    /* db2[i] = (STUDENT_TYPE*) malloc(sizeof(
        STUDENT_TYPE)); */
    db2[i] = malloc (sizeof (STUDENT_TYPE));
    db2[i]->id = 200 + i;
    db2[i]->grade = 'a' + rand () % 5;
    strcpy (db2[i]->name, "John Doe");
    printf ("%d %c %s\t\t", db2[i]->id, db2[i]->grade,
        db2[i]->name);
    printf ("%d %c %s\n", (*db2[i]).id, (*db2[i]).grade
        , (*db2[i]).name);
  }
  for (i = 0; i < MAX; i++) {
    free (db2[i]);
  }


  return 0;
}
```

ex7-1.c

```
/* code: ex7-1.c   (v1.20.00) */
#include <stdio.h>
#include <stdlib.h>
#define ARRAY_SIZE 13

/* ------------------------------------------------ */
int linear_search (int array[], int n, int key)
{
  int i;
  for (i = 0; i < n; i++) {
    if (array[i] == key) {
      return i;
    }
  }
  return -1;
}

/* ------------------------------------------------ */
void print_array (int array[], int n)
{
  int i;
  for (i = 0; i < n; i++) {
    printf ("%d ", array[i]);
  }
  printf ("\n");
}

/* ------------------------------------------------ */
int main ()
{
  int index, key;
  int array[ARRAY_SIZE] = {
    900, 990, 210, 50, 80, 150, 330,
    470, 510, 530, 800, 250, 280
  };
  key = 800;
  print_array (array, ARRAY_SIZE);
  index = linear_search (array, ARRAY_SIZE, key);
  if (index != -1) {
    printf ("Found: %d (Index:%d)\n", key, index);
  }
  else {
    printf ("Not found: %d\n", key);
  }
  return 0;
}
```

ex7-2.c

```c
/* code: ex7-2.c    (v1.20.00) */
#include <stdio.h>
#include <stdlib.h>
#define ARRAY_SIZE 13

/* ------------------------------------------------------------ */
int binary_search (int array[], int num, int key)
{
  int middle, low, high;
  low = 0;
  high = num - 1;
  while (low <= high) {
    middle = (low + high) / 2;
    if (key == array[middle]) {
      return middle;
    }
    else if (key < array[middle]) {
      high = middle - 1;
    }
    else {
      low = middle + 1;
    }
  }
  return -1;
}

/* ------------------------------------------------------------ */
void print_array (int array[], int n)
{
  int i;
  for (i = 0; i < n; i++) {
    printf ("%d ", array[i]);
  }
  printf ("\n");
}
```

```

/* ---------------------------------------------------------------- */
int main ()
{
  int index, key;
  int array[ARRAY_SIZE] = {
    50, 80, 150, 210, 250, 280, 330,
    470, 510, 530, 800, 900, 990
  };

  key = 800;
  print_array (array, ARRAY_SIZE);
  index = binary_search (array, ARRAY_SIZE, key);
  if (index != -1) {
    printf ("Found: %d (Index:%d)\n", key, index);
  }
  else {
    printf ("Not found: %d\n", key);
  }
  return 0;
}
```

q7-1.c

```
/* code: q7-1.c    (v1.25.00) */
#include <stdio.h>
#include <stdlib.h>
#include <search.h>

#define ARRAY_SIZE 5
#define EXTRA_ROOM 1

/* ------------------------------------------------ */
int compare (const void *x, const void *y)
{

int a = *(const int *) x;

int b = *(const int *) y;

return (a > b) - (a < b);

}



/* ------------------------------------------------ */
void print_array (int arr[], size_t n)
{

for (size_t i = 0; i < n; i++) {

printf ("%d ", arr[i]);

}
printf ("\n");

}

```

```

/* ----------------------------------------------------------- */
int main ()
{

int key = 25;

size_t elements = ARRAY_SIZE;

int array[ARRAY_SIZE + EXTRA_ROOM] = { 10, 20, 30, 40,
    50 };


printf ("Initial array: ");

print_array (array, elements);


int *found = lsearch (&key, array, &elements, sizeof (
    int), compare);


if (!found) {

printf ("Search error\n");

return EXIT_FAILURE;

}


printf ("\nResult:\n");

if (elements == ARRAY_SIZE) {

printf ("Key %d was already in the array.\n", key);

}
```

```
  else {

printf ("Key %d was added to the array.\n", key);

}


print_array (array, elements);

return EXIT_SUCCESS;

}
```

q7-2.c

```c
/* code: q7-2.c    (v1.25.00) */
#include <stdio.h>
#include <stdlib.h>
#include <search.h>

#define ARRAY_SIZE 10

/* ------------------------------------------------ */
int compare (const void *a, const void *b)
{
  int x = *(int *) a;
  int y = *(int *) b;
  if (x < y)
    return -1;
  if (x > y)
    return 1;
  return 0;
}

/* ------------------------------------------------ */
void print_array (int array[], int n)
{
  int i;
  for (i = 0; i < n; i++) {
    printf ("%d ", array[i]);
  }
  printf ("\n");
}

/* ------------------------------------------------ */
int main ()
{
  int key;
  int *r;
  int array[ARRAY_SIZE] = {
    10, 12, 16, 19, 28, 30, 38, 44, 70, 98
  };                                /* ordered array! */

  key = 16;
  print_array (array, ARRAY_SIZE);

  r = (int *) bsearch (&key, array, ARRAY_SIZE, sizeof
     (int), compare);
  if (r != NULL) {
    printf ("Found: %d\n", *r);
  }
  else {
    printf ("Not found: %d\n", key);
  }
  return 0;
}
```

q7-3.c

```c
/* code: q7-3.c    (v1.25.00) */

#include <stdio.h>
#include <stdlib.h>
#define MAX 1000000

/* ------------------------------------------------------------ */
void array_print (int a[], int max)
{
  int i;
  for (i = 0; i < max; i++) {
    printf ("%02d ", a[i]);
  }
  printf ("\n");
}

/* ------------------------------------------------------------ */
int array_find_empty (int a[], int max)
{
  int i;
  for (i = 0; i < max; i++) {
    if (a[i] == -1) {
      return i;
    }
  }
  return -1;
}

/* ------------------------------------------------------------ */
void array_insert (int a[], int max, int index, int
   empty, int data)
{
  int i;
  if (empty > index) {
    for (i = empty; i > index; i--) {
      a[i] = a[i - 1];
    }
  }
  else {
    for (i = empty; i < index; i++) {
      a[i] = a[i + 1];
    }
  }
  a[index] = data;
}

/* ------------------------------------------------------------ */
int array_delete (int a[], int index)
{
  int data;
  data = a[index];
  a[index] = -1;
  return data;
}


/* ------------------------------------------------------------ */
int main ()
{
  int i, j, index_ins, index_del, empty, data;

  int *a;
  a = (int *) malloc (sizeof (int) * MAX);
  if (a == NULL) {
    printf ("Error! memory not allocated.");
    exit (EXIT_FAILURE);
  }

  for (j = 0; j < MAX; j++) {
    a[j] = rand () % 100;
  }
  data = a[MAX / 2];
  a[MAX / 2] = -1;

  for (i = 0; i < 1000; i++) {
    index_ins = rand () % MAX;
    index_del = rand () % MAX;
    /* printf("ins:%d  del:%d\n", index_ins, index_del
       );  */

    empty = array_find_empty (a, MAX);
    /* printf("empty:%d\n", empty ); */

    array_insert (a, MAX, index_ins, empty, data);

    data = array_delete (a, index_del);
    /* array_print( a, MAX ); */
  }


  free (a);

  return 0;
}

/* ---------------------------------------------------- */
/*
 $ gcc -pg -Wall q7-3.c -o q7-3
 $ ./q7-3
 $ gprof q7-3 gmon.out > gmon.log
 $ more gmon.log
 */
/* ---------------------------------------------------- */
```

ex8-1.c

```c
/* code: ex8-1.c    (v1.20.00) */
#include<stdio.h>
#include<stdlib.h>

#define MAX 128
#define PUSH_SUCCESS      1
#define PUSH_FAILURE     -1
#define POP_SUCCESS       2
#define POP_FAILURE      -2

/* ------------------------------------------------------------ */
void stack_init (int *top)
{
  *top = 0;
}

/* ------------------------------------------------------------ */
void display (int stack[], int top)
{
  int i;
  printf ("STACK(%d): ", top);
  for (i = 0; i < top; i++) {
    printf ("%d ", stack[i]);
  }
  printf ("\n");
}

/* ------------------------------------------------------------ */
int push (int stack[], int *top, int data)
{
  if (*top >= MAX) {
    /* stack overflow */
    return PUSH_FAILURE;
  }
  else {
    stack[*top] = data;
    (*top)++;
    return PUSH_SUCCESS;
  }
}

/* ---------------------------------------------------------------- */
int pop (int stack[], int *top, int *data)
{
  if ((*top) > 0) {
    *data = stack[(*top) - 1];
    (*top)--;
    return POP_SUCCESS;
  }
  else {
    /* stack empty */
    return POP_FAILURE;
  }
}

/* ---------------------------------------------------------------- */
int main ()
{
  int stack[MAX];
  int top, data;

  stack_init (&top);
  data = 300;
  printf ("push: %d\n", data);
  push (stack, &top, data);
  data = 400;
  printf ("push: %d\n", data);
  push (stack, &top, data);
  data = 500;
  printf ("push: %d\n", data);
  push (stack, &top, data);
  display (stack, top);
  pop (stack, &top, &data);
  printf ("pop: %d\n", data);
  pop (stack, &top, &data);
  printf ("pop: %d\n", data);
  pop (stack, &top, &data);
  printf ("pop: %d\n", data);
  return 0;
}
```

ex8-2.c

```
/* code: ex8-2.c    (v1.20.00) */
#include<stdio.h>
#include<stdlib.h>

#define MAX 128
#define ENQUEUE_SUCCESS       1
#define ENQUEUE_FAILURE      -1
#define DEQUEUE_SUCCESS       2
#define DEQUEUE_FAILURE      -2

/* ---------------------------------------------- */
void queue_init (int *front, int *rear)
{
  *front = -1;
  *rear = -1;
}

/* ---------------------------------------------- */
int enqueue (int q[], int *rear, int data)
{
  if (*rear < MAX - 1) {
    *rear = *rear + 1;
    q[*rear] = data;
    return ENQUEUE_SUCCESS;
  }
  else {
    return ENQUEUE_FAILURE;
  }
}

/* ---------------------------------------------- */
int dequeue (int q[], int *front, int rear, int *data)
{
  if (*front == rear) {
    return DEQUEUE_FAILURE;
  }
  *front = *front + 1;
  *data = q[*front];
  return DEQUEUE_SUCCESS;
}

/* ---------------------------------------------------------- */
int main ()
{
  int queue[MAX];
  int front, rear, data;
  int stat;

  queue_init (&front, &rear);
  enqueue (queue, &rear, 100);
  enqueue (queue, &rear, 200);
  enqueue (queue, &rear, 300);
  enqueue (queue, &rear, 400);
  enqueue (queue, &rear, 500);
  while (rear - front) {
    stat = dequeue (queue, &front, rear, &data);
    if (stat == DEQUEUE_SUCCESS) {
      printf ("%d\n", data);
    }
    else {
      printf ("QUEUE is empty\n");
    }
  }
  return 0;
}
```

ex8-3.c

```c
/* code: ex8-3.c    (v1.20.00) */
#include <stdio.h>
#include <stdlib.h>
#define WIDTH  40
#define HEIGHT 20

/* ------------------------------------------------------------ */
void cell_evolve (int array[HEIGHT][WIDTH])
{
  int array_new[HEIGHT][WIDTH];
  int x, y, n, x_width, y_height;

  for (y = 0; y < HEIGHT; y++) {
    for (x = 0; x < WIDTH; x++) {
      n = 0;
      for (y_height = y - 1; y_height <= y + 1;
          y_height++) {
        for (x_width = x - 1; x_width <= x + 1; x_width
            ++) {
          if (array[(y_height + HEIGHT) % HEIGHT][(
              x_width + WIDTH) % WIDTH]) {
            n++;
          }
        }
      }
      if (array[y][x]) {
        n--;
      }
      array_new[y][x] = (n == 3 || (n == 2 && array[y][
          x]));
    }
  }

  for (y = 0; y < HEIGHT; y++) {
    for (x = 0; x < WIDTH; x++) {
            array[y][x] = array_new[y][x];
        }
    }
}

/* -------------------------------------------------- */
void cell_first_generation (int array[HEIGHT][WIDTH])
{
    int x, y, r;
    for (x = 0; x < WIDTH; x++) {
        for (y = 0; y < HEIGHT; y++) {
            r = RAND_MAX / 8;
            if (rand () < r) {
                array[y][x] = 1;
            }
            else {
                array[y][x] = 0;
            }
        }
    }
}

/* -------------------------------------------------- */
void cell_print (int array[HEIGHT][WIDTH], int
    generation)
{
    int x, y;
    printf ("[Generation: %05d]\n", generation);
    for (y = 0; y < HEIGHT; y++) {
        for (x = 0; x < WIDTH; x++) {
            if (array[y][x] == 1) {
                printf ("*");
            }
            else {
                printf (".");
            }
        }
        printf ("\n");
```

```
  }
  printf ("\n");
  fflush (stdout);
}

/* ---------------------------------------------------------------- */
int main ()
{
  int i;
  int array[HEIGHT][WIDTH];
  cell_first_generation (array);
  i = 0;
  while (i < 100) {
    cell_print (array, i);
    cell_evolve (array);
    i++;
  }
  return 0;
}
```

q8-1.c

```
/* code: q8-1.c    (v1.20.00) */
#include<stdio.h>
#include<stdlib.h>

#define MAX 128                        /* ring buffer max size
     */

#define ENQUEUE_SUCCESS     1
#define ENQUEUE_FAILURE    -1
#define DEQUEUE_SUCCESS     2
#define DEQUEUE_FAILURE    -2

/* ------------------------------------------------ */
void rb_queue_init (int *front, int *rear)
{
  *front = 0;
  *rear = 0;
}

/* ------------------------------------------------ */
int rb_enqueue (int q[], int *front, int *rear, int
   data)
{
  int index_f, index_r, index_q;
  index_f = *front % MAX;
  index_r = (*rear + 1) % MAX;
  if (index_f != index_r) {
    index_q = (*rear)++ % MAX;
    q[index_q] = data;
    return ENQUEUE_SUCCESS;
  }
  else {
    return ENQUEUE_FAILURE;
  }
}
```

```

/* --------------------------------------------------- */
int rb_dequeue (int q[], int *front, int *rear, int *
  data)
{
  int index;
  if (*front != *rear) {
    index = (*front)++ % MAX;
    *data = q[index];
    return DEQUEUE_SUCCESS;
  }
  else {
    return DEQUEUE_FAILURE;
  }
}

/* --------------------------------------------------- */
int main ()
{
  int queue[MAX];
  int front, rear, data;
  int stat;

  rb_queue_init (&front, &rear);

  rb_enqueue (queue, &front, &rear, 100);
  rb_enqueue (queue, &front, &rear, 200);
  rb_enqueue (queue, &front, &rear, 300);
  rb_enqueue (queue, &front, &rear, 400);
  rb_enqueue (queue, &front, &rear, 500);

  while (rear - front) {
    stat = rb_dequeue (queue, &front, &rear, &data);
    if (stat == DEQUEUE_SUCCESS) {
      printf ("%d\n", data);
    }
    else {
      printf ("QUEUE is empty\n");
```

```
        }
    }

    return 0;
}
```

q8-2.c

```
/* code: q8-2.c    (v1.20.00) */
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#define WIDTH  40
#define HEIGHT 20

/* ------------------------------------------------ */
void cell_evolve (int array[HEIGHT][WIDTH])
{
  int array_new[HEIGHT][WIDTH];
  int x, y, n, x_width, y_height;

  for (y = 0; y < HEIGHT; y++) {
    for (x = 0; x < WIDTH; x++) {
      n = 0;
      for (y_height = y - 1; y_height <= y + 1;
          y_height++) {
        for (x_width = x - 1; x_width <= x + 1; x_width
            ++) {
          if (array[(y_height + HEIGHT) % HEIGHT][(
              x_width + WIDTH) % WIDTH]) {
            n++;
          }
        }
      }
      if (array[y][x]) {
        n--;
      }
      array_new[y][x] = (n == 3 || (n == 2 && array[y][
          x]));
    }
  }

  for (y = 0; y < HEIGHT; y++) {
    for (x = 0; x < WIDTH; x++) {
      array[y][x] = array_new[y][x];
    }
  }
}

/* ---------------------------------------------------------- */
void cell_first_generation (int array[HEIGHT][WIDTH])
{
  int x, y, r;
  for (x = 0; x < WIDTH; x++) {
    for (y = 0; y < HEIGHT; y++) {
      r = RAND_MAX / 8;
      if (rand () < r) {
        array[y][x] = 1;
      }
      else {
        array[y][x] = 0;
      }
    }
  }
}


/* ---------------------------------------------------------- */
void cell_print (int array[HEIGHT][WIDTH], int
   generation)
{
  int x, y;

  printf ("[Generation: %05d]\n", generation);
  for (y = 0; y < HEIGHT; y++) {
    for (x = 0; x < WIDTH; x++) {
      if (array[y][x] == 1) {
        printf ("*");
      }
      else {
        printf (".");
      }
    }
    printf ("\n");
  }
  printf ("\n");
  fflush (stdout);
}


/* ------------------------------------------------------------ */
void cell_print_esc (int array[HEIGHT][WIDTH], int
   generation)
{
  int x, y;

  printf ("\e[H");
  for (y = 0; y < HEIGHT; y++) {
    for (x = 0; x < WIDTH; x++) {
      if (array[y][x] == 1) {
        printf ("\e[07m  \e[m");
      }
      else {
        printf ("  ");
      }
    }
    printf ("\e[E");
  }
  fflush (stdout);
}

/* ------------------------------------------------------------ */
int main ()
{
  int i;
  int array[HEIGHT][WIDTH];

  cell_first_generation (array);
  i = 0;
  while (i < 100) {
    cell_print_esc (array, i);
    cell_evolve (array);
    i++;
    sleep (1);
  }

  return 0;
}
```

ex9-1.c

```
/* code: ex9-1.c   (v1.20.00) */
#include <stdio.h>
#include <stdlib.h>

int main ()
{
  FILE *fptr;
  fptr = fopen ("ex9-1-output.txt", "w");
  fprintf (fptr, "The Open University of Japan\n");
  fclose (fptr);
  return 0;
}
```

ex9-2.c

```c
/* code: ex9-2.c   (v1.20.00) */
#include <stdio.h>
#include <stdlib.h>

int main ()
{
  FILE *fptr;
  if (NULL == (fptr = fopen ("ex9-2-output.txt", "w"))) {
    fprintf (stderr, "ERROR: Can not open file [output2.txt]");
    exit (-1);
  }
  fprintf (fptr, "The Open University of Japan\n");
  fclose (fptr);
  return 0;
}
```

ex9-3.c

```c
/* code: ex9-3.c    (v1.20.00) */
#include <stdio.h>
#include <stdlib.h>

#define IRIS_DATA "iris.dat"

int main ()
{
  FILE *fptr;
  float sl, sw, pl, pw;
  float s_sl, s_sw, s_pl, s_pw;
  char name[128];
  int n;

  if (NULL == (fptr = fopen (IRIS_DATA, "r"))) {
    fprintf (stderr, "ERROR: Can not open file [%s]",
      IRIS_DATA);
    exit (-1);
  }
  n = 0;
  s_sl = s_sw = s_pl = s_pw = 0.0;
  while (EOF != fscanf (fptr, "%f,%f,%f,%f,%s", &sl, &
     sw, &pl, &pw, name)) {
    s_sl += sl;
    s_sw += sw;
    s_pl += pl;
    s_pw += pw;
    n++;
  }
  printf ("iris data : %d\n", n);
  printf ("avg. sepal length: %f\n", s_sl / (float) n);
  printf ("avg. sepal width : %f\n", s_sw / (float) n);
  printf ("avg. petal length: %f\n", s_pl / (float) n);
  printf ("avg. petal width : %f\n", s_pw / (float) n);
  fclose (fptr);

  return 0;
}
```

q9-1.c

```
/* code: q9-1.c    (v1.20.00) */
#include <stdio.h>
#include <stdlib.h>

#define IRIS_DATA "iris.dat"
#define MAX_ARRAY 256

struct iris
{
  float sl, sw, pl, pw;
  char name[128];
};
typedef struct iris IRIS_TYPE;

/* ------------------------------------------------------------ */
int read_iris_data (IRIS_TYPE data[], int num, char *
    filename)
{
  FILE *fptr;
  float sl, sw, pl, pw;
  char name[128];
  int n;
  if (NULL == (fptr = fopen (filename, "r"))) {
    fprintf (stderr, "ERROR: Can not open file [%s]",
        filename);
    exit (-1);
  }
  n = 0;
  while (EOF != fscanf (fptr, "%f,%f,%f,%f,%s", &sl, &
      sw, &pl, &pw, name)) {
    data[n].sl = sl;
    data[n].sw = sw;
    data[n].pl = pl;
    data[n].pw = pw;
    sprintf (data[n].name, "%s", name);
```

```
      n++;
      if (n >= num) {
        fprintf (stderr, "ERROR: Not enough array size");
        exit (-1);
      }
    }
    fclose (fptr);
    return n;
}

/* ------------------------------------------------------------ */
IRIS_TYPE find_iris_avg (IRIS_TYPE data[], int num)
{
    IRIS_TYPE avg;
    int i;
    float s_sl, s_sw, s_pl, s_pw;

    s_sl = s_sw = s_pl = s_pw = 0.0;
    for (i = 0; i < num; i++) {
      s_sl += data[i].sl;
      s_sw += data[i].sw;
      s_pl += data[i].pl;
      s_pw += data[i].pw;
    }

    avg.sl = s_sl / (float) num;
    avg.sw = s_sw / (float) num;
    avg.pl = s_pl / (float) num;
    avg.pw = s_pw / (float) num;
    sprintf (avg.name, "%s", "average");
    return avg;
}

/* ------------------------------------------------------------ */
int main ()
{
    int num;
    IRIS_TYPE iris_data[MAX_ARRAY];
```

```
  IRIS_TYPE avg;

  num = read_iris_data (iris_data, MAX_ARRAY, IRIS_DATA);
  avg = find_iris_avg (iris_data, num);
  printf ("iris data : %d\n", num);
  printf ("avg. sepal length: %f\n", avg.sl);
  printf ("avg. sepal width : %f\n", avg.sw);
  printf ("avg. petal length: %f\n", avg.pl);
  printf ("avg. petal width : %f\n", avg.pw);

  return 0;
}
```

q9-2.c

```c
/* code: q9-2.c    (v1.20.00) */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define ROW  256
#define COL  256
#define FILTER_SIZE 3

struct pgm
{
  int row;
  int col;
  int max;
  float pixel[ROW][COL];
};
typedef struct pgm PGM_TYPE;


/* ------------------------------------------------------------ */
void pgm_read (PGM_TYPE *image, char *filename)
{
  FILE *infile;
  int i, j;
  char magic_number[32];

  if (NULL == (infile = fopen (filename, "r"))) {
    fprintf (stderr, "Can-not-open-file-[%s]", filename
        );
    exit (-1);
  }
  fscanf (infile, "%s", magic_number);
  fscanf (infile, "%d", &image->col);
  fscanf (infile, "%d", &image->row);
  fscanf (infile, "%d", &image->max);
  printf ("image:%s␣[%dx%d]␣(%d)\n",
          magic_number, image->col, image->row, image->max);
  if (strcmp ("P2", magic_number)) {
    fprintf (stderr, "Not␣PGM(P2)␣file!");
    exit (-2);
  }
  for (i = 0; i < image->row; i++) {
    for (j = 0; j < image->col; j++) {
      fscanf (infile, "%f", &image->pixel[i][j]);
    }
  }
  fclose (infile);
}

/* -------------------------------------------------------- */
void pgm_write (PGM_TYPE *image, char *filename)
{
  FILE *outfile;
  int i, j;

  if (NULL == (outfile = fopen (filename, "w"))) {
    fprintf (stderr, "Can␣not␣open␣file␣[%s]", filename);
    exit (-1);
  }
  fprintf (outfile, "%s\n", "P2");
  fprintf (outfile, "%d␣", image->col);
  fprintf (outfile, "%d␣\n", image->row);
  fprintf (outfile, "%d␣\n", image->max);

  for (i = 0; i < image->row; i++) {
    for (j = 0; j < image->col; j++) {
      fprintf (outfile, "%2d␣", (int) image->pixel[i][j]);
    }
    fprintf (outfile, "\n");
  }
  fclose (outfile);
}

/* ------------------------------------------------ */
void convolution (PGM_TYPE *image_input, PGM_TYPE *
    image_output)
{
  int i, j, k, l, px, py, sum;
  float dsum;

  float filter[FILTER_SIZE][FILTER_SIZE] = {
    {-1.0, -1.0, -1.0},
    {-1.0, +8.0, -1.0},
    {-1.0, -1.0, -1.0}
  };

  image_output->row = image_input->row;
  image_output->col = image_input->col;
  image_output->max = image_input->max;
  px = (FILTER_SIZE - 1) / 2;
  py = (FILTER_SIZE - 1) / 2;
  dsum = 0;
  for (i = 0 + px; i < image_input->row - px; i++) {
    for (j = 0 + py; j < image_input->col - py; j++) {
      dsum = 0.0;
      for (k = 0; k < FILTER_SIZE; k++) {
        for (l = 0; l < FILTER_SIZE; l++) {
          dsum += filter[k][l] * image_input->pixel[i -
              px + k][j - py + l];
        }
      }
      sum = (int) dsum;
      if (sum > image_input->max) {
        sum = image_input->max;
      }
      if (sum < 0) {
        sum = 0;
      }
      image_output->pixel[i][j] = sum;
    }
  }
}

/* ------------------------------------------------------------ */
int main ()
{
  PGM_TYPE *pgm_input;
  PGM_TYPE *pgm_output;

  pgm_input = malloc (sizeof (PGM_TYPE));
  pgm_output = malloc (sizeof (PGM_TYPE));

  pgm_read (pgm_input, "sample.pgm");
  convolution (pgm_input, pgm_output);
  pgm_write (pgm_output, "q9-2-laplacian8.pgm");

  free (pgm_input);
  free (pgm_output);

  return 0;
}
```

q9-3.c

```c
/* code: q9-3.c    (v1.20.00) */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define ROW  256
#define COL  256
#define FILTER_SIZE 3

struct pgm
{
  int row;
  int col;
  int max;
  float pixel[ROW][COL];
};
typedef struct pgm PGM_TYPE;


/* ------------------------------------------------ */
void pgm_read (PGM_TYPE *image, char *filename)
{
  FILE *infile;
  int i, j;
  char magic_number[32];

  if (NULL == (infile = fopen (filename, "r"))) {
    fprintf (stderr, "Can not open file [%s]", filename
      );
    exit (-1);
  }
  fscanf (infile, "%s", magic_number);
  fscanf (infile, "%d", &image->col);
  fscanf (infile, "%d", &image->row);
  fscanf (infile, "%d", &image->max);
  printf ("image:%s [%dx%d] (%d)\n",
          magic_number, image->col, image->row, image->max);
  if (strcmp ("P2", magic_number)) {
    fprintf (stderr, "Not PGM(P2) file!");
    exit (-2);
  }
  for (i = 0; i < image->row; i++) {
    for (j = 0; j < image->col; j++) {
      fscanf (infile, "%f", &image->pixel[i][j]);
    }
  }
  fclose (infile);
}

/* ---------------------------------------------------------- */
void pgm_write (PGM_TYPE *image, char *filename)
{
  FILE *outfile;
  int i, j;

  if (NULL == (outfile = fopen (filename, "w"))) {
    fprintf (stderr, "Can not open file [%s]", filename);
    exit (-1);
  }
  fprintf (outfile, "%s\n", "P2");
  fprintf (outfile, "%d ", image->col);
  fprintf (outfile, "%d \n", image->row);
  fprintf (outfile, "%d \n", image->max);

  for (i = 0; i < image->row; i++) {
    for (j = 0; j < image->col; j++) {
      fprintf (outfile, "%2d ", (int) image->pixel[i][j]);
    }
    fprintf (outfile, "\n");
  }
  fclose (outfile);
}

/* ---------------------------------------------------- */
void convolution (PGM_TYPE *image_input, PGM_TYPE *
    image_output)
{
  int i, j, k, l, px, py, sum;
  float dsum;

  float filter[FILTER_SIZE][FILTER_SIZE] = {
    {0.0, -1.0, -2.0},
    {1.0, 0.0, -1.0},
    {2.0, 1.0, 0.0}
  };

  image_output->row = image_input->row;
  image_output->col = image_input->col;
  image_output->max = image_input->max;
  px = (FILTER_SIZE - 1) / 2;
  py = (FILTER_SIZE - 1) / 2;
  dsum = 0;
  for (i = 0 + px; i < image_input->row - px; i++) {
    for (j = 0 + py; j < image_input->col - py; j++) {
      dsum = 0.0;
      for (k = 0; k < FILTER_SIZE; k++) {
        for (l = 0; l < FILTER_SIZE; l++) {
          dsum += filter[k][l] * image_input->pixel[i -
              px + k][j - py + l];
        }
      }
      sum = (int) dsum;
      if (sum > image_input->max) {
        sum = image_input->max;
      }
      if (sum < 0) {
        sum = 0;
      }
```

```
        image_output->pixel[i][j] = sum;
      }
    }
}

/* ---------------------------------------------------------------- */
int main ()
{
  PGM_TYPE *pgm_input;
  PGM_TYPE *pgm_output;

  pgm_input = malloc (sizeof (PGM_TYPE));
  pgm_output = malloc (sizeof (PGM_TYPE));

  pgm_read (pgm_input, "sample.pgm");
  convolution (pgm_input, pgm_output);
  pgm_write (pgm_output, "q9-3-sobel-d.pgm");

  free (pgm_input);
  free (pgm_output);

  return 0;
}
```

q9-4.c

```c
/* code: q9-4.c    (v1.20.00) */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define ROW  256
#define COL  256
#define FILTER_SIZE 3

struct pgm
{
  int row;
  int col;
  int max;
  float pixel[ROW][COL];
};
typedef struct pgm PGM_TYPE;


/* ------------------------------------------------------------ */
void pgm_read (PGM_TYPE *image, char *filename)
{
  FILE *infile;
  int i, j;
  char magic_number[32];

  if (NULL == (infile = fopen (filename, "r"))) {
    fprintf (stderr, "Can-not-open-file-[%s]", filename
        );
    exit (-1);
  }
  fscanf (infile, "%s", magic_number);
  fscanf (infile, "%d", &image->col);
  fscanf (infile, "%d", &image->row);
  fscanf (infile, "%d", &image->max);
  printf ("image:%s [%dx%d] (%d)\n",
          magic_number, image->col, image->row, image->max);
  if (strcmp ("P2", magic_number)) {
    fprintf (stderr, "Not PGM(P2) file!");
    exit (-2);
  }
  for (i = 0; i < image->row; i++) {
    for (j = 0; j < image->col; j++) {
      fscanf (infile, "%f", &image->pixel[i][j]);
    }
  }
  fclose (infile);
}

/* ---------------------------------------------------------------- */
void pgm_write (PGM_TYPE *image, char *filename)
{
  FILE *outfile;
  int i, j;

  if (NULL == (outfile = fopen (filename, "w"))) {
    fprintf (stderr, "Can not open file [%s]", filename);
    exit (-1);
  }
  fprintf (outfile, "%s\n", "P2");
  fprintf (outfile, "%d ", image->col);
  fprintf (outfile, "%d \n", image->row);
  fprintf (outfile, "%d \n", image->max);

  for (i = 0; i < image->row; i++) {
    for (j = 0; j < image->col; j++) {
      fprintf (outfile, "%2d ", (int) image->pixel[i][j]);
    }
    fprintf (outfile, "\n");
  }
  fclose (outfile);
}

/* ---------------------------------------------- */
void convolution (PGM_TYPE *image_input, PGM_TYPE *
    image_output)
{
  int i, j, k, l, px, py, sum;
  float dsum;

  float filter[FILTER_SIZE][FILTER_SIZE] =
    { {1.00 / 16.00, 2.00 / 16.00, 1.00 / 16.00},
  {2.00 / 16.00, 4.00 / 16.00, 2.00 / 16.00},
  {1.00 / 16.00, 2.00 / 16.00, 1.00 / 16.00}
  };

  image_output->row = image_input->row;
  image_output->col = image_input->col;
  image_output->max = image_input->max;
  px = (FILTER_SIZE - 1) / 2;
  py = (FILTER_SIZE - 1) / 2;
  dsum = 0;
  for (i = 0 + px; i < image_input->row - px; i++) {
    for (j = 0 + py; j < image_input->col - py; j++) {
      dsum = 0.0;
      for (k = 0; k < FILTER_SIZE; k++) {
        for (l = 0; l < FILTER_SIZE; l++) {
          dsum += filter[k][l] * image_input->pixel[i -
              px + k][j - py + l];
        }
      }
      sum = (int) dsum;
      if (sum > image_input->max) {
        sum = image_input->max;
      }
      if (sum < 0) {
        sum = 0;
      }
```

```
            image_output->pixel[i][j] = sum;
        }
    }
}

/* ---------------------------------------------------------------------- */
int main ()
{
  PGM_TYPE *pgm_input;
  PGM_TYPE *pgm_output;

  pgm_input = malloc (sizeof (PGM_TYPE));
  pgm_output = malloc (sizeof (PGM_TYPE));

  pgm_read (pgm_input, "sample.pgm");
  convolution (pgm_input, pgm_output);
  pgm_write (pgm_output, "q9-4-gaussian.pgm");

  free (pgm_input);
  free (pgm_output);

  return 0;
}
```

ex10-1.c

```
/* code: ex10-1.c    (v1.20.00) */
#include <stdio.h>
#include <stdlib.h>
/* ------------------------------------------------ */
void print_array (int v[], int n)
{
  int i;
  printf ("array: ");
  for (i = 0; i < n; i++) {
    printf ("%d ", v[i]);
  }
  printf ("\n");
}

/* ------------------------------------------------ */
void bubble_sort (int v[], int n)
{
  int i, j, t;
  for (i = 0; i < n - 1; i++) {
    for (j = n - 1; j > i; j--) {
      if (v[j - 1] > v[j]) {
        t = v[j];
        v[j] = v[j - 1];
        v[j - 1] = t;
      }
      printf ("i:%d j:%d  ", i, j);
      print_array (v, n);
    }
  }
}

/* ------------------------------------------------ */
int main ()
{
  int array[5]
```

```
    = { 30, 50, 20, 10, 40 };
  print_array (array, 5);
  bubble_sort (array, 5);
  print_array (array, 5);
  return 0;
}
```

ex10-2.c

```c
/* code: ex10-2.c   (v1.20.00) */
#include <stdio.h>
#include <stdlib.h>

/* ------------------------------------------------ */
void print_array (int v[], int n)
{
  int i;
  printf ("array: ");
  for (i = 0; i < n; i++) {
    printf ("%d ", v[i]);
  }
  printf ("\n");
}

/* ------------------------------------------------ */
void selection_sort (int v[], int n)
{
  int i, j, t, min_index;
  for (i = 0; i < n - 1; i++) {
    min_index = i;
    for (j = i + 1; j < n; j++) {
      if (v[j] < v[min_index]) {
        min_index = j;
      }
      printf ("i:%d j:%d  ", i, j);
      print_array (v, n);
    }
    t = v[i];
    v[i] = v[min_index];
    v[min_index] = t;
  }
}

/* ------------------------------------------------ */
```

```
int main ()
{
  int array[5]
  = { 30, 50, 20, 10, 40 };
  print_array (array, 5);
  selection_sort (array, 5);
  print_array (array, 5);
  return 0;
}
```

ex10-3.c

```c
/* code: ex10-3.c    (v1.20.00) */
#include <stdio.h>
#include <stdlib.h>

/* ------------------------------------------------ */
void print_array (int v[], int n)
{
  int i;
  printf ("array:␣");
  for (i = 0; i < n; i++) {
    printf ("%d␣", v[i]);
  }
  printf ("\n");
}

/* ------------------------------------------------ */
void insertion_sort (int v[], int n)
{
  int i, j, t;
  for (i = 1; i < n; i++) {
    j = i;
    while ((j >= 1) && (v[j - 1] > v[j])) {
      t = v[j];
      v[j] = v[j - 1];
      v[j - 1] = t;
      j--;
      printf ("i:%d␣j:%d␣␣", i, j);
      print_array (v, n);
    }
  }
}

/* ------------------------------------------------ */
int main ()
{
  int array[5]
  = { 30, 50, 20, 10, 40 };
  print_array (array, 5);
  insertion_sort (array, 5);
  print_array (array, 5);
  return 0;
}
```

ex10-4.c

```c
/* code: ex10-4.c    (v1.20.00) */
#include <stdio.h>
#include <stdlib.h>

/* ------------------------------------------------------------ */
void print_array (int v[], int n)
{
  int i;
  printf ("array:␣");
  for (i = 0; i < n; i++) {
    printf ("%d␣", v[i]);
  }
  printf ("\n");
}

/* ------------------------------------------------------------ */
void insertion_sort (int v[], int n)
{
  int i, j, t;
  for (i = 1; i < n; i++) {
    j = i;
    while ((j >= 1) && (v[j - 1] > v[j])) {
      t = v[j];
      v[j] = v[j - 1];
      v[j - 1] = t;
      j--;
      printf ("i:%d␣j:%d␣␣", i, j);
      print_array (v, n);
    }
  }
}

/* ------------------------------------------------------------ */
int main ()
{
  int array[5]
  = { 10, 20, 30, 40, 50 };
  print_array (array, 5);
  insertion_sort (array, 5);
  print_array (array, 5);
  return 0;
}
```

ex10-5.c

```
/* code: ex10-5.c    (v1.20.00) */
#include <stdio.h>
#include <stdlib.h>

/* ------------------------------------------------ */
void print_array (int v[], int n)
{
  int i;
  printf ("array: ");
  for (i = 0; i < n; i++) {
    printf ("%d ", v[i]);
  }
  printf ("\n");
}

/* ------------------------------------------------ */
void insertion_sort (int v[], int n)
{
  int i, j, t;
  for (i = 1; i < n; i++) {
    j = i;
    while ((j >= 1) && (v[j - 1] > v[j])) {
      t = v[j];
      v[j] = v[j - 1];
      v[j - 1] = t;
      j--;
      printf ("i:%d j:%d  ", i, j);
      print_array (v, n);
    }
  }
}

/* ------------------------------------------------ */
int main ()
{
  int array[5] = { 50, 40, 30, 20, 10 };
  print_array (array, 5);
  insertion_sort (array, 5);
  print_array (array, 5);
  return 0;
}
```

q10-1.c

```c
/* code: q10-1.c    (v1.20.00) */
#include <stdio.h>
#include <stdlib.h>

/* ------------------------------------------------ */
void print_array (int v[], int n)
{
  int i;
  printf ("array: ");
  for (i = 0; i < n; i++) {
    printf ("%d ", v[i]);
  }
  printf ("\n");
}

/* ------------------------------------------------ */
void bubble_sort (int v[], int n)
{
  int i, j, t, flag;

  flag = 1;
  for (i = 0; (i < n - 1) && (flag == 1); i++) {
    flag = 0;
    for (j = n - 1; j > i; j--) {
      if (v[j - 1] > v[j]) {
        t = v[j];
        v[j] = v[j - 1];
        v[j - 1] = t;
        flag = 1;
      }
      printf ("i:%d j:%d  ", i, j);
      print_array (v, n);
    }
  }
}
```

```

/* -------------------------------------------------- */
int main ()
{
/*  int array[5]
      = { 30, 50, 20, 10, 40 }; */
/*  int array[5]
      = { 50, 40, 30, 20, 10 }; */

  int array[5]
  = { 10, 20, 30, 40, 50 };

  print_array (array, 5);
  bubble_sort (array, 5);
  print_array (array, 5);

  return 0;
}
```

q10-2.c

```
/* code: q10-2.c    (v1.20.00) */
#include <stdio.h>
#include <stdlib.h>

/* ------------------------------------------------------------ */
void print_array (int v[], int n)
{
  int i;
  printf ("array: ");
  for (i = 0; i < n; i++) {
    printf ("%d ", v[i]);
  }
  printf ("\n");
}

/* ------------------------------------------------------------ */
int int_compare (const void *va, const void *vb)
{
  int a, b;
  a = *(int *) va;
  b = *(int *) vb;
  if (a < b) {
    return (-1);
  }
  else if (a > b) {
    return (1);
  }
  else {
    return (0);
  }
}

/* ------------------------------------------------------------ */
int main ()
{
  int array[5]
  = { 30, 50, 20, 10, 40 };

  print_array (array, 5);
  qsort (array, 5, sizeof (int), int_compare);
  print_array (array, 5);

  return 0;
}
```

q10-3.c

```c
/* code: q10-3.c    (v1.20.00) */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* ------------------------------------------------ */
int cmp_string (const void *p1, const void *p2)
{
  return strcmp (*(char *const *) p1, *(char *const *)
      p2);
}

/* ------------------------------------------------ */
void print_str_array (char *v[], int n)
{
  int i;
  printf ("array: ");
  for (i = 0; i < n; i++) {
    printf ("%s ", v[i]);
  }
  printf ("\n");
}

/* ------------------------------------------------ */
int main ()
{
  char *array[7] = {
    "Sunday", "Monday", "Tuesday",
    "Wednesday", "Thursday", "Friday", "Saturday",
  };
  print_str_array (array, 7);
  qsort (array, 7, sizeof (char *), cmp_string);
  print_str_array (array, 7);
  return 0;
}
```

ex11-1.c

```
/* code: ex11-1.c   (v1.20.00) */
#include <stdio.h>
#include <stdlib.h>

/* ------------------------------------------ */
void print_array (int v[], int n)
{
  int i;
  printf ("array: ");
  for (i = 0; i < n; i++) {
    printf ("%d ", v[i]);
  }
  printf ("\n");
}

/* ------------------------------------------ */
int partition (int v[], int lower_bound, int upper_bound)
{
  int a, down, up, temp;

  a = v[lower_bound];
  up = upper_bound;
  down = lower_bound;

  while (down < up) {
    while ((v[down] <= a) && (down < upper_bound)) {
      down++;
    }
    while (v[up] > a) {
      up--;
    }
    if (down < up) {
      temp = v[down];
      v[down] = v[up];
```

```
        v[up] = temp;
      }
    }
    v[lower_bound] = v[up];
    v[up] = a;
    return up;
}

/* ------------------------------------------------------------ */
void quicksort (int v[], int left, int right)
{
    int p;
    if (left >= right) {
      return;
    }
    p = partition (v, left, right);
    quicksort (v, left, p - 1);
    quicksort (v, p + 1, right);
}

/* ------------------------------------------------------------ */
int main ()
{
    int array[10]
    = { 80, 40, 30, 20, 10, 00, 70, 90, 50, 60 };

    print_array (array, 10);
    quicksort (array, 0, 9);
    print_array (array, 10);

    return 0;
}
```

ex11-2.c

```c
/* code: ex11-2.c    (v1.20.00) */
#include <stdio.h>
#include <stdlib.h>
#define MAX 10

/* ------------------------------------------------ */
void print_array (int v[], int n)
{
  int i;
  printf ("array: ");
  for (i = 0; i < n; i++) {
    printf ("%4d ", v[i]);
  }
  printf ("\n");
}

/* ------------------------------------------------ */
void radix_sort (int a[], int n)
{
  int i, max, exp;
  int temp[MAX];
  int bucket[10];

  max = 0;
  exp = 1;
  for (i = 0; i < n; i++) {
    if (a[i] > max) {
      max = a[i];
    }
  }
  while (max / exp > 0) {
    for (i = 0; i < 10; i++) {
      bucket[i] = 0;
    }
    for (i = 0; i < n; i++) {
      bucket[a[i] / exp % 10]++;
    }
    for (i = 1; i < 10; i++) {
      bucket[i] += bucket[i - 1];
    }
    for (i = n - 1; i >= 0; i--) {
      temp[--bucket[a[i] / exp % 10]] = a[i];
    }
    for (i = 0; i < n; i++) {
      a[i] = temp[i];
    }
    exp *= 10;
    print_array (a, n);
  }
}

/* ------------------------------------------------------------ */
int main ()
{
  int array[MAX]
  = { 12, 19, 10, 28, 30, 01, 502, 16, 34, 177 };

  print_array (array, 10);
  radix_sort (array, 10);
  print_array (array, 10);

  return 0;
}
```

q11-1.c

```
/* code: q11-1.c    (v1.20.00) */
#include <stdio.h>
#include <stdlib.h>

/* ------------------------------------------------ */
void print_array (int v[], int n)
{
  int i;
  printf ("array: ");
  for (i = 0; i < n; i++) {
    printf ("%d ", v[i]);
  }
  printf ("\n");
}

/* ------------------------------------------------ */
int partition (int v[], int lower_bound, int upper_bound)
{
  int a, down, up, temp;

  a = v[lower_bound];
  up = upper_bound;
  down = lower_bound;

  while (down < up) {
    while ((v[down] <= a) && (down < upper_bound)) {
      down++;
    }
    while (v[up] > a) {
      up--;
    }
    if (down < up) {
      temp = v[down];
      v[down] = v[up];
      v[up] = temp;
    }
  }
  v[lower_bound] = v[up];
  v[up] = a;
  return up;
}

/* ------------------------------------------------------------ */
void quicksort (int v[], int left, int right, int level
  )
{
  int p;
  int i;
  if (left >= right) {
    return;
  }
  level++;
  p = partition (v, left, right);
  quicksort (v, left, p - 1, level);
  quicksort (v, p + 1, right, level);
  for (i = 0; i < level; i++) {
    printf ("*");
  }
  printf ("[pivot:%d left:%d right:%d]\n", p, left,
     right);
}

/* ------------------------------------------------------------ */
int main ()
{
  int array[10]
  = { 30, 40, 80, 20, 10, 00, 70, 90, 50, 60 };

  print_array (array, 10);
  quicksort (array, 0, 9, 0);
  print_array (array, 10);

```

```
    return 0;
}
```

q11-2.c

```
/* code: q11-2.c    (v1.20.00) */
#include <stdio.h>
#include <stdlib.h>
#define STACK_SIZE 2048

/* ------------------------------------------------------------ */
void print_array (int v[], int n)
{
  int i;
  printf ("\n");
  for (i = 0; i < n; i++) {
    printf ("%02d ", v[i]);
  }
}

/* ------------------------------------------------------------ */
int partition (int v[], int lower_bound, int
    upper_bound)
{
  int a, down, up, temp;

  a = v[lower_bound];
  up = upper_bound;
  down = lower_bound;

  while (down < up) {
    while ((v[down] <= a) && (down < upper_bound)) {
      down++;
    }
    while (v[up] > a) {
      up--;
    }
    if (down < up) {
      temp = v[down];
      v[down] = v[up];
      v[up] = temp;
    }
  }
  v[lower_bound] = v[up];
  v[up] = a;
  return up;
}

/* ---------------------------------------------------------- */
void quicksort_stack (int v[], int n)
{
  int left, right, i, sptr;
  int stack_lower_bound[STACK_SIZE];
  int stack_upper_bound[STACK_SIZE];

  stack_lower_bound[0] = 0;
  stack_upper_bound[0] = n - 1;
  sptr = 1;

  while (sptr > 0) {
    sptr--;
    left = stack_lower_bound[sptr];
    right = stack_upper_bound[sptr];

    if (left >= right) {
      ;
    }
    else {
      i = partition (v, left, right);

      if ((i - left) < (right - i)) {
        stack_lower_bound[sptr] = i + 1;
        stack_upper_bound[sptr++] = right;
        stack_lower_bound[sptr] = left;
        stack_upper_bound[sptr++] = i - 1;
      }
      else {
        stack_lower_bound[sptr] = left;
```

```
          stack_upper_bound[sptr++] = i - 1;
          stack_lower_bound[sptr] = i + 1;
          stack_upper_bound[sptr++] = right;
        }
      }
    }
}


/* ------------------------------------------------------------ */
int main ()
{
  int array[10]
  = { 80, 40, 30, 20, 10, 00, 70, 90, 50, 60 };

  print_array (array, 10);
  quicksort_stack (array, 10);
  print_array (array, 10);

  return 0;
}
```

q11-3.c

```
/* code: q11-3.c    (v1.20.00) */
#include <stdio.h>
#include <stdlib.h>

#define MAX 100000000

/* ------------------------------------------------------------ */
void print_array (int v[], int n)
{
  int i;
  printf ("array: ");
  for (i = 0; i < n; i++) {
    printf ("%8d ", v[i]);
  }
  printf ("\n");
}

/* ------------------------------------------------------------ */
void rand_data (int v[], int n)
{
  int i;
  for (i = 0; i < n; i++) {
    v[i] = rand () % (MAX / 10);
  }
}

/* ------------------------------------------------------------ */
int partition (int v[], int lower_bound, int
    upper_bound)
{
  int a, down, up, temp;

  a = v[lower_bound];
  up = upper_bound;
  down = lower_bound;

  while (down < up) {
    while ((v[down] <= a) && (down < upper_bound)) {
      down++;
    }
    while (v[up] > a) {
      up--;
    }
    if (down < up) {
      temp = v[down];
      v[down] = v[up];
      v[up] = temp;
    }
  }
  v[lower_bound] = v[up];
  v[up] = a;
  return up;
}

/* ------------------------------------------------ */
void quicksort (int v[], int left, int right)
{
  int p;
  if (left >= right) {
    return;
  }
  p = partition (v, left, right);
  quicksort (v, left, p - 1);
  quicksort (v, p + 1, right);
}

/* ------------------------------------------------ */
int main (int argc, char **argv)
{
  int *array;

  array = malloc (sizeof (int) * MAX);
  if (array == NULL) {
    printf ("ERROR:␣Can␣not␣allocate␣memory\n");
    exit (-1);
  }

  rand_data (array, MAX);
  printf ("array␣size:␣%d\n", MAX);
  fflush (stdout);

  print_array (array, 30);

  quicksort (array, 0, MAX - 1);

  print_array (array, 30);

  free (array);
  return 0;
}
```

q11-4.c

```c
/* code: q11-4.c    (v1.20.00) */
#include <stdio.h>
#include <stdlib.h>

/* ------------------------------------------------------------ */
void print_array (int v[], int n)
{
  int i;
  printf ("\n");
  for (i = 0; i < n; i++) {
    printf ("%d ", v[i]);
  }
}

/* ------------------------------------------------------------ */
void merge_sort (int v[], int lb, int ub, int v_temp[])
{
  int i, j, k, c;

  if (lb >= ub) {
    return;
  }
  c = (lb + ub) / 2;

  merge_sort (v, lb, c, v_temp);
  merge_sort (v, c + 1, ub, v_temp);

  for (i = lb; i <= c; i++) {
    v_temp[i] = v[i];
  }
  for (i = c + 1, j = ub; i <= ub; i++, j--) {
    v_temp[i] = v[j];
  }

  i = lb;
  j = ub;

  for (k = lb; k <= ub; k++) {
    if (v_temp[i] <= v_temp[j]) {
      v[k] = v_temp[i++];
    }
    else {
      v[k] = v_temp[j--];
    }
  }
}

/* ------------------------------------------------------------ */
int main ()
{
  int array[10] = { 8, 4, 3, 2, 1, 0, 7, 9, 5, 6 };
  int array_temp[10];

  print_array (array, 10);
  merge_sort (array, 0, 9, array_temp);
  print_array (array, 10);

  return 0;
}
```

ex12-1.c

```
/* code: ex12-1.c    (v1.20.00) */
#include <stdio.h>
#include <stdlib.h>
#define ARRAY_SIZE 3000000

int main ()
{
  int array[ARRAY_SIZE];
  int i;
  for (i = 0; i < ARRAY_SIZE; i++) {
    array[i] = 100;
  }
  for (i = 0; i < 10; i++) {
    printf ("%d ", array[i]);
  }
  return 0;
}
```

ex12-2.c

```
/* code: ex12-2.c    (v1.20.00) */
#include <stdio.h>
#include <stdlib.h>
#define ARRAY_SIZE 3000000

int array[ARRAY_SIZE];

int main ()
{
  int i;
  for (i = 0; i < ARRAY_SIZE; i++) {
    array[i] = 100;
  }
  for (i = 0; i < 10; i++) {
    printf ("%d ", array[i]);
  }
  return 0;
}
```

ex12-3.c

```c
/* code: ex12-3.c    (v1.20.00) */
#include <stdio.h>
#include <stdlib.h>
#define ARRAY_SIZE 3000000

int main ()
{
  int *array;
  int i;

  array = malloc (sizeof (int) * ARRAY_SIZE);

  if (NULL == array) {
    fprintf (stderr, "Error: malloc() \n");
    exit (-1);
  }
  else {
    for (i = 0; i < ARRAY_SIZE; i++) {
      array[i] = 100;
    }
    for (i = 0; i < 10; i++) {
      printf ("%d ", array[i]);
    }
    free (array);
  }

  return 0;
}
```

ex12-4.c

```c
/* code: ex12-4.c   (v1.20.00) */
#include <stdio.h>
#include <stdlib.h>

/* ------------------------------------------------ */
int main ()
{
  int **array;
  int i, j, rows, columns;

  rows = 768;
  columns = 1024;

  array = malloc (rows * sizeof (int *));
  for (i = 0; i < rows; i++) {
    array[i] = malloc (columns * sizeof (int));
  }

  for (i = 0; i < rows; i++) {
    for (j = 0; j < columns; j++) {
      array[i][j] = rand () % 10;
    }
  }

  for (i = 0; i < rows; i++) {
    for (j = 0; j < columns; j++) {
      printf ("%d ", array[i][j]);
    }
    printf ("\n");
  }

  for (i = 0; i < rows; i++) {
    free (array[i]);
  }
  free (array);

  return 0;
}
```

ex12-5.c

```
/* code: ex12-5.c   (v1.20.00) */
#include <stdio.h>
#include <stdlib.h>

/* ------------------------------------------------------------ */
int main ()
{
  int **array;
  int i, j, rows, columns;

  rows = 768;
  columns = 1024;

  array = malloc (rows * sizeof (int *));
  array[0] = malloc (rows * columns * sizeof (int));
  for (i = 1; i < rows; i++) {
    array[i] = array[0] + i * columns;
  }

  for (i = 0; i < rows; i++) {
    for (j = 0; j < columns; j++) {
      array[i][j] = rand () % 10;
    }
  }

  for (i = 0; i < rows; i++) {
    for (j = 0; j < columns; j++) {
      printf ("%d ", array[i][j]);
    }
    printf ("\n");
  }

  free (array[0]);
  free (array);

```

```
    return 0;
}
```

q12-1.c

```c
/* code: q12-1.c    (v1.20.00) */
#include <stdio.h>
#include <stdlib.h>

/* ------------------------------------------------------------ */
int main ()
{
  int ***array;
  int i, j, k;
  int x, y, z;

  x = 10;
  y = 20;
  z = 30;

  array = malloc (x * sizeof (int **));
  for (i = 0; i < x; i++) {
    array[i] = malloc (y * sizeof (int *));
    for (j = 0; j < y; j++) {
      array[i][j] = malloc (z * sizeof (int));
    }
  }

  for (i = 0; i < x; i++) {
    for (j = 0; j < y; j++) {
      for (k = 0; k < z; k++) {
        array[i][j][k] = rand () % 10;
      }
    }
  }

  for (i = 0; i < x; i++) {
    for (j = 0; j < y; j++) {
      for (k = 0; k < z; k++) {
        printf ("%d ", array[i][j][k]);
```

```
            }
        }
    }

    for (i = 0; i < x; i++) {
        for (j = 0; j < y; j++) {
            free (array[i][j]);
        }
        free (array[i]);
    }
    free (array);

    return 0;
}
```

q12-2.c

```
/* code: q12-2.c    (v1.20.00) */
#include <stdio.h>
#include <stdlib.h>

/* ------------------------------------------ */
int **m2d_allocate (int rows, int columns)
{
  int **array;
  int i;
  array = malloc (rows * sizeof (int *));
  for (i = 0; i < rows; i++) {
    array[i] = malloc (columns * sizeof (int));
  }
  return array;
}

/* ------------------------------------------ */
void m2d_deallocate (int **array, int rows)
{
  int i;
  for (i = 0; i < rows; i++) {
    free (array[i]);
  }
  free (array);
}

/* ------------------------------------------ */
int main ()
{
  int **array;
  int i, j, rows, columns;

  rows = 768;
  columns = 1024;

  array = m2d_allocate (rows, columns);

  for (i = 0; i < rows; i++) {
    for (j = 0; j < columns; j++) {
      array[i][j] = rand () % 10;
    }
  }

  for (i = 0; i < rows; i++) {
    for (j = 0; j < columns; j++) {
      printf ("%d ", array[i][j]);
    }
    printf ("\n");
  }

  m2d_deallocate (array, rows);

  return 0;
}
```

ex13-1.c

```c
/* code: ex13-1.c    (v1.20.00) */
#include<stdio.h>
#include<stdlib.h>

struct node
{
  int data;
  struct node *next;
};
typedef struct node NODE_TYPE;

/* ------------------------------------------ */
void linked_list_print (NODE_TYPE *node)
{
  while (NULL != node) {
    printf ("%d-", node->data);
    node = node->next;
  }
  printf ("\n");
}

/* ------------------------------------------ */
int main ()
{
  NODE_TYPE *node;
  node = malloc (sizeof (NODE_TYPE));
  node->data = 300;
  node->next = malloc (sizeof (NODE_TYPE));
  node->next->data = 400;
  node->next->next = malloc (sizeof (NODE_TYPE));
  node->next->next->data = 500;
  node->next->next->next = malloc (sizeof (NODE_TYPE));
  node->next->next->next->data = 600;
  node->next->next->next->next = NULL;
  linked_list_print (node);
```

```
    return 0;
}
```

ex13-2.c

```c
/* code: ex13-2.c   (v1.20.00) */
#include<stdio.h>
#include<stdlib.h>

#define NOT_FOUND (-1)
#define DATA_SIZE 6

struct node
{
  int data;
  struct node *next;
};
typedef struct node NODE_TYPE;

/* ------------------------------------------ */
void linked_list_insert_head (NODE_TYPE **head, int
    data)
{
  NODE_TYPE *new_node;
  new_node = malloc (sizeof (NODE_TYPE));
  new_node->data = data;
  if (*head == NULL) {
    new_node->next = NULL;
    *head = new_node;
  }
  else {
    new_node->next = *head;
    *head = new_node;
  }
}

/* ------------------------------------------ */
void linked_list_print (NODE_TYPE *head)
{
  printf ("Linked list [ ");
  while (NULL != head) {
    printf ("%02d ", head->data);
    head = head->next;
  }
  printf ("]\n");
}

/* ------------------------------------------------------------ */
int main ()
{
  NODE_TYPE *head;
  int i, data1;

  head = NULL;
  for (i = 0; i < DATA_SIZE; i++) {
    data1 = (int) rand () % 100;
    printf ("inserting (head): ");
    printf ("%02d\n", data1);
    linked_list_insert_head (&head, data1);
  }
  linked_list_print (head);
  return 0;
}
```

ex13-3.c

```c
/* code: ex13-3.c   (v1.20.00) */
#include<stdio.h>
#include<stdlib.h>

#define NOT_FOUND (-1)
#define DATA_SIZE 6

struct node
{
  int data;
  struct node *next;
};
typedef struct node NODE_TYPE;

/* ------------------------------------------------ */
int linked_list_delete_head (NODE_TYPE **head)
{
  int data;
  NODE_TYPE *temp;
  if (*head == NULL) {
    return NOT_FOUND;
  }
  data = (*head)->data;
  temp = (*head);
  *head = (*head)->next;
  free (temp);
  return data;
}

/* ------------------------------------------------ */
void linked_list_insert_head (NODE_TYPE **head, int
    data)
{
  NODE_TYPE *new_node;
  new_node = malloc (sizeof (NODE_TYPE));
  new_node->data = data;
  if (*head == NULL) {
    new_node->next = NULL;
    *head = new_node;
  }
  else {
    new_node->next = *head;
    *head = new_node;
  }
}

/* ------------------------------------------------------------ */
void linked_list_print (NODE_TYPE *head)
{
  printf ("Linked list [ ");
  while (NULL != head) {
    printf ("%02d ", head->data);
    head = head->next;
  }
  printf ("]\n");
}

/* ------------------------------------------------------------ */
int main ()
{
  NODE_TYPE *head;
  int i, data1;

  head = NULL;
  for (i = 0; i < DATA_SIZE; i++) {
    data1 = (int) rand () % 100;
    printf ("inserting (head): ");
    printf ("%02d\n", data1);
    linked_list_insert_head (&head, data1);
  }
  linked_list_print (head);
  for (i = 0; i < DATA_SIZE / 2; i++) {
    printf ("deleting (head): ");
    data1 = linked_list_delete_head (&head);
    printf ("%02d\n", data1);
  }
  linked_list_print (head);
  return 0;
}
```

ex13-4.c

```c
/* code: ex13-4.c    (v1.20.00) */
#include<stdio.h>
#include<stdlib.h>

#define NOT_FOUND (-1)
#define DATA_SIZE 6

struct node
{
  int data;
  struct node *next;
};
typedef struct node NODE_TYPE;

/* ------------------------------------------------------------ */
int linked_list_search_node (NODE_TYPE *head, int key)
{
  int i;
  i = 0;
  while (NULL != head) {
    if (key == head->data) {
      return i;
    }
    head = head->next;
    i++;
  }
  return NOT_FOUND;
}

/* ------------------------------------------------------------ */
int linked_list_length (NODE_TYPE *head)
{
  int c;
  c = 0;
  while (NULL != head) {
    head = head->next;
    c++;
  }
  return c;
}

/* -------------------------------------------------------- */
void linked_list_print (NODE_TYPE *head)
{
  printf ("Linked_list [ ");
  while (NULL != head) {
    printf ("%02d ", head->data);
    head = head->next;
  }
  printf ("]\n");
}

/* -------------------------------------------------------- */
void linked_list_insert_head (NODE_TYPE **head, int data)
{
  NODE_TYPE *new_node;
  new_node = malloc (sizeof (NODE_TYPE));
  new_node->data = data;
  if (*head == NULL) {
    new_node->next = NULL;
    *head = new_node;
  }
  else {
    new_node->next = *head;
    *head = new_node;
  }
}

/* -------------------------------------------------------- */
int linked_list_delete_head (NODE_TYPE **head)
{
  int data;
  NODE_TYPE *temp;
  if (*head == NULL) {
    return NOT_FOUND;
  }
  data = (*head)->data;
  temp = (*head);
  *head = (*head)->next;
  free (temp);
  return data;
}

/* ----------------------------------------------------------- */
void linked_list_insert_tail (NODE_TYPE **head, int
    data)
{
  NODE_TYPE *new_node;
  NODE_TYPE *temp;
  new_node = malloc (sizeof (NODE_TYPE));

  new_node->data = data;
  new_node->next = NULL;
  if (*head == NULL) {
    *head = new_node;
    temp = new_node;
  }
  else {
    temp = *head;
    while (temp->next != NULL) {
      temp = temp->next;
    }
    temp->next = new_node;
  }
}

/* ----------------------------------------------------------- */
int linked_list_delete_tail (NODE_TYPE **head)
{
  int data;
  NODE_TYPE *temp;
  NODE_TYPE *prev;

  data = NOT_FOUND;
  if (*head == NULL) {
    return data;
  }
  else {
    temp = *head;
    prev = *head;
    while (temp->next != NULL) {
      prev = temp;
      temp = temp->next;
    }
    data = temp->data;
    if ((*head)->next == NULL) {
      *head = NULL;
    }
    else {
      prev->next = NULL;
    }
    free (temp);
  }
  return data;
}


/* ------------------------------------------------------------ */
void linked_list_delete_all (NODE_TYPE **head)
{
  NODE_TYPE *current;
  NODE_TYPE *next;

  current = *head;
  while (current != NULL) {
    next = current->next;
    free (current);
    current = next;
  }
  *head = NULL;
}


/* ------------------------------------------------------------ */
int main ()
{
  NODE_TYPE *head;
  int i, data1;

  head = NULL;
  for (i = 0; i < DATA_SIZE; i++) {
    data1 = (int) rand () % 100;
    printf ("adding (head node): ");
    printf ("%02d\n", data1);
    linked_list_insert_head (&head, data1);
  }
  linked_list_print (head);

  for (i = 0; i < DATA_SIZE; i++) {
    data1 = (int) rand () % 100;
    printf ("adding (tail node): ");
    printf ("%02d\n", data1);
    linked_list_insert_tail (&head, data1);
  }
  linked_list_print (head);

  for (i = 0; i < DATA_SIZE / 2; i++) {
    printf ("deleting (head node): ");
    data1 = linked_list_delete_head (&head);
    printf ("%02d\n", data1);
  }
  linked_list_print (head);

  for (i = 0; i < DATA_SIZE / 2; i++) {
    printf ("deleting (tail node): ");
    data1 = linked_list_delete_tail (&head);
    printf ("%02d\n", data1);
  }
  linked_list_print (head);
  printf ("Length of the linked list :%d\n",
     linked_list_length (head));

  printf ("deleting entire linked list\n");
  linked_list_delete_all (&head);
  linked_list_print (head);
  return 0;
}
```

q13-1.c

```c
/* code: q13-1.c    (v1.20.00) */
#include<stdio.h>
#include<stdlib.h>

#define NOT_FOUND (-1)
#define DATA_SIZE 6

struct node
{
  int data;
  struct node *next;
};
typedef struct node NODE_TYPE;

/* ------------------------------------------------ */
int linked_list_search_node (NODE_TYPE *head, int key)
{
  int i;
  i = 0;
  while (NULL != head) {
    if (key == head->data) {
      return i;
    }
    head = head->next;
    i++;
  }
  return NOT_FOUND;
}

/* ------------------------------------------------ */
void linked_list_print (NODE_TYPE *head)
{
  printf ("Linked list [ ");
  while (NULL != head) {
    printf ("%02d ", head->data);
```

```
      head = head->next;
    }
    printf ("]\n");
}


/* ---------------------------------------------------------------- */
void linked_list_insert_head (NODE_TYPE **head, int
    data)
{
   NODE_TYPE *new_node;
   new_node = malloc (sizeof (NODE_TYPE));
   new_node->data = data;
   if (*head == NULL) {
      new_node->next = NULL;
      *head = new_node;
   }
   else {
      new_node->next = *head;
      *head = new_node;
   }
}

/* ---------------------------------------------------------------- */
int main ()
{
   NODE_TYPE *head;
   int i, data1, data2, stat;

   head = NULL;
   data2 = 0;
   for (i = 0; i < DATA_SIZE; i++) {
      data1 = (int) rand () % 100;
      printf ("adding (head node): ");
      printf ("%02d\n", data1);
      linked_list_insert_head (&head, data1);
      if (i == (DATA_SIZE / 2)) {
         data2 = data1;
      }
   }

   linked_list_print (head);

   stat = linked_list_search_node (head, data2);
   if (stat == NOT_FOUND) {
      printf ("not found : %d", data2);
   }
   else {
      printf ("found : %d (%d)", data2, stat);
   }
   return 0;
}
```

q13-2.c

```c
/* code: q13-2.c    (v1.20.00) */
#include<stdio.h>
#include<stdlib.h>

#define NOT_FOUND (-1)
#define DATA_SIZE 6

struct node
{
  int data;
  struct node *next;
};
typedef struct node NODE_TYPE;

/* ------------------------------------------------------------ */
int linked_list_length (NODE_TYPE *head)
{
  int c;
  c = 0;
  while (NULL != head) {
    head = head->next;
    c++;
  }
  return c;
}

/* ------------------------------------------------------------ */
void linked_list_print (NODE_TYPE *head)
{
  printf ("Linked_list [ ");
  while (NULL != head) {
    printf ("%02d ", head->data);
    head = head->next;
  }
  printf ("]\n");
```

```
}


/* ------------------------------------------------------------ */
void linked_list_insert_head (NODE_TYPE **head, int
    data)
{
  NODE_TYPE *new_node;
  new_node = malloc (sizeof (NODE_TYPE));
  new_node->data = data;
  if (*head == NULL) {
    new_node->next = NULL;
    *head = new_node;
  }
  else {
    new_node->next = *head;
    *head = new_node;
  }
}

/* ------------------------------------------------------------ */
int main ()
{
  NODE_TYPE *head;
  int i, data1;

  head = NULL;
  for (i = 0; i < DATA_SIZE; i++) {
    data1 = (int) rand () % 100;
    printf ("adding (head node): ");
    printf ("%02d\n", data1);
    linked_list_insert_head (&head, data1);
  }
  linked_list_print (head);
  printf ("number of node(s): %d\n", linked_list_length
      (head));

  return 0;
```

```
}
```

q13-3.c

```c
/* code: q13-3.c    (v1.20.00) */
#include<stdio.h>
#include<stdlib.h>

#define NOT_FOUND (-1)
#define DATA_SIZE 6

struct node
{
  int data;
  struct node *next;
};
typedef struct node NODE_TYPE;

/* ------------------------------------------------ */
void linked_list_print (NODE_TYPE *head)
{
  printf ("Linked_list [ ");
  while (NULL != head) {
    printf ("%02d ", head->data);
    head = head->next;
  }
  printf ("]\n\n");
}


/* ------------------------------------------------ */
void linked_list_insert_tail (NODE_TYPE **head, int
    data)
{
  NODE_TYPE *new_node;
  NODE_TYPE *temp;
  new_node = malloc (sizeof (NODE_TYPE));

  new_node->data = data;
  new_node->next = NULL;
  if (*head == NULL) {
    *head = new_node;
    temp = new_node;
  }
  else {
    temp = *head;
    while (temp->next != NULL) {
      temp = temp->next;
    }
    temp->next = new_node;
  }
}

/* ---------------------------------------------------------------- */
int linked_list_delete_tail (NODE_TYPE **head)
{
  int data;
  NODE_TYPE *temp;
  NODE_TYPE *prev;

  data = NOT_FOUND;
  if (*head == NULL) {
    return data;
  }
  else {
    temp = *head;
    prev = *head;
    while (temp->next != NULL) {
      prev = temp;
      temp = temp->next;
    }
    data = temp->data;
    if ((*head)->next == NULL) {
      *head = NULL;
    }
    else {
      prev->next = NULL;
    }
    free (temp);
  }
  return data;
}



/* ------------------------------------------------------------ */
int main ()
{
  NODE_TYPE *head;
  int i, data1;

  head = NULL;
  for (i = 0; i < DATA_SIZE; i++) {
    data1 = (int) rand () % 100;
    printf ("adding (tail node): ");
    printf ("%02d\n", data1);
    linked_list_insert_tail (&head, data1);
  }


  linked_list_print (head);

  for (i = 0; i < DATA_SIZE; i++) {
    printf ("deleting (tail node): ");
    data1 = linked_list_delete_tail (&head);
    printf ("%02d\n", data1);
  }

  linked_list_print (head);
  return 0;
}
```

q13-4.c

```
/* code: q13-4.c   (v1.20.00) */
#include<stdio.h>
#include<stdlib.h>

#define DATA_SIZE 10

struct node
{
  int data;
  struct node *next;
};
typedef struct node NODE_TYPE;

struct linked_list
{
  NODE_TYPE *head;
  NODE_TYPE *tail;
  NODE_TYPE *current;
};
typedef struct linked_list LINKED_LIST;

/* ------------------------------------------------ */
void linked_list_init (LINKED_LIST *list)
{
  list->head = NULL;
  list->tail = NULL;
  list->current = NULL;
}


/* ------------------------------------------------ */
void linked_list_insert_node_h (LINKED_LIST *list, int
    data)
{
  NODE_TYPE *node;
```

```c

  node = malloc (sizeof (NODE_TYPE));

  node->data = data;

  if (NULL == list->head) {
    list->tail = node;
    node->next = NULL;
  }
  else {
    node->next = list->head;
  }
  list->head = node;
}

/* ------------------------------------------------ */
void linked_list_insert_node_t (LINKED_LIST *list, int data)
{
  NODE_TYPE *node;

  node = malloc (sizeof (NODE_TYPE));

  node->data = data;
  node->next = NULL;

  if (NULL == list->head) {
    list->head = node;
  }
  else {
    list->tail->next = node;
  }
  list->tail = node;
}

/* ------------------------------------------------ */
NODE_TYPE *linked_list_find_node (LINKED_LIST *list, int data)
```

```
{
  NODE_TYPE *node;

  node = list->head;
  while (NULL != node) {
    if (node->data == data) {
      return node;
    }
    node = node->next;
  }
  return NULL;
}


/* ------------------------------------------------ */
void linked_list_delete_node (LINKED_LIST *list ,
    NODE_TYPE *node)
{
  if (node == list->head) {
    if (NULL == list->head->next) {
      list->head = NULL;
      list->tail = NULL;
    }
    else {
      list->head = list->head->next;
    }
  }
  else {
    NODE_TYPE *temp;
    temp = list->head;
    while ((NULL != temp) && (temp->next != node)) {
      temp = temp->next;
    }
    if (NULL != temp) {
      temp->next = node->next;
    }
  }
  free (node);
```

```
}

/* ---------------------------------------------------------------- */
void linked_list_print (LINKED_LIST *list)
{
  NODE_TYPE *node;

  printf ("linked list [ ");
  node = list->head;
  while (NULL != node) {
    printf ("%02d ", node->data);
    node = node->next;
  }
  printf ("]\n");
}

/* ---------------------------------------------------------------- */
int linked_list_count_node (LINKED_LIST *list)
{
  NODE_TYPE *node;
  int i;

  i = 0;
  node = list->head;
  while (NULL != node) {
    i++;
    node = node->next;
  }
  return i;
}



/* ---------------------------------------------------------------- */
int main ()
{
  LINKED_LIST *list;
  NODE_TYPE *node;
  int i, data1, data2, del_data;

  list = malloc (sizeof (LINKED_LIST));
  linked_list_init (list);

  for (i = 0; i < DATA_SIZE; i++) {
    data1 = (int) rand () % 100;
    printf ("adding node to head of linked list: ");
    printf ("%02d\n", data1);
    linked_list_insert_node_h (list, data1);
  }
  linked_list_print (list);


  for (i = 0; i < DATA_SIZE; i++) {
    data2 = (int) rand () % 100;
    printf ("adding node to tail of linked list: ");
    printf ("%02d\n", data2);
    linked_list_insert_node_t (list, data2);
  }
  linked_list_print (list);

  del_data = data2;
  printf ("finding node:  [%d]\n", del_data);
  node = linked_list_find_node (list, del_data);
  if (NULL != node) {
    printf ("deleting node: [%d]\n", del_data);
    linked_list_delete_node (list, node);
  }
  else {
    printf ("Node not found: [%d]\n", del_data);
  }
  linked_list_print (list);

  printf ("number of node(s):%d\n",
     linked_list_count_node (list));

  free (list);
```

```

    return 0;
}
```

ex14-1.c

```c
/* code: ex14-1.c    (v1.20.00) */
#include<stdio.h>
#include<stdlib.h>

#define STACK_UNDERFLOW (-1)
#define DATA_SIZE 6

struct node
{
  int data;
  struct node *next;
};
typedef struct node NODE_TYPE;


/* ------------------------------------------------------------ */
void stack_push (NODE_TYPE **head, int data)
{
  NODE_TYPE *new_node;
  new_node = malloc (sizeof (NODE_TYPE));
  new_node->data = data;
  new_node->next = *head;
  *head = new_node;
}

/* ------------------------------------------------------------ */
int stack_pop (NODE_TYPE **head)
{
  int data;
  NODE_TYPE *temp;
  if (*head == NULL) {
    return STACK_UNDERFLOW;
  }
  data = (*head)->data;
  temp = (*head);
  *head = (*head)->next;
  free (temp);
  return data;
}

/* ------------------------------------------------------------ */
void stack_print (NODE_TYPE *head)
{
  if (head == NULL) {
    printf ("stack is empty.\n");
    return;
  }
  printf ("stack [ ");
  while (NULL != head) {
    printf ("%02d ", head->data);
    head = head->next;
  }
  printf ("]\n");
}

/* ------------------------------------------------------------ */
int main ()
{
  NODE_TYPE *stack;
  int i, data1;
  stack = NULL;
  for (i = 0; i < DATA_SIZE; i++) {
    data1 = (int) rand () % 100;
    printf ("push: ");
    printf ("%02d\n", data1);
    stack_push (&stack, data1);
  }
  stack_print (stack);
  for (i = 0; i < DATA_SIZE / 2; i++) {
    printf ("pop: ");
    data1 = stack_pop (&stack);
    printf ("%02d\n", data1);
  }
  stack_print (stack);
  return 0;
}
```

ex14-2.c

```
/* code: ex14-2.c   (v1.20.00) */
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define DATA_SIZE 6
#define QUEUE_EMPTY (-1)

struct node
{
  int data;
  struct node *next;
};
typedef struct node NODE_TYPE;

/* ------------------------------------------------------------ */
void q_enque (NODE_TYPE **front, NODE_TYPE **rear, int
    data)
{
  NODE_TYPE *new_node;
  new_node = malloc (sizeof (NODE_TYPE));
  new_node->data = data;
  new_node->next = NULL;
  if (*rear == NULL) {
    *front = *rear = new_node;
  }
  else {
    (*rear)->next = new_node;
    *rear = new_node;
  }
}

/* ------------------------------------------------------------ */
int q_dequeue (NODE_TYPE **front, NODE_TYPE **rear)
{
  int data;
  NODE_TYPE *temp;
  if (*front == NULL) {
    return QUEUE_EMPTY;
  }
  temp = *front;
  data = (*front)->data;
  if (*front == *rear) {
    *front = *rear = NULL;
  }
  else {
    *front = (*front)->next;
  }
  free (temp);
  return data;
}

/* ------------------------------------------------------------ */
void q_print (NODE_TYPE *front)
{
  printf ("queue␣[␣");
  while (front != NULL) {
    printf ("%02d␣", front->data);
    front = front->next;
  }
  printf ("]\n");
}

/* ------------------------------------------------------------ */
int main ()
{
  int i, data1;
  NODE_TYPE *front, *rear;

  front = NULL;
  rear = NULL;
  for (i = 0; i < DATA_SIZE; i++) {
    data1 = (int) rand () % 100;
    printf ("enqueue: ");
    printf ("%02d\n", data1);
    q_enque (&front, &rear, data1);
  }
  q_print (front);
  for (i = 0; i < DATA_SIZE / 2; i++) {
    printf ("dequeue: ");
    data1 = q_dequeue (&front, &rear);
    printf ("%02d\n", data1);
  }
  q_print (front);

  return 0;
}
```

q14-1.c

```
/* code: q14-1.c    (v1.20.00) */

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

#define DATA_SIZE 6

/* doubly linked list */
struct node
{
  int data;
  struct node *prev;
  struct node *next;
};
typedef struct node NODE_TYPE;

/* ------------------------------------------------------------ */
void dll_print_head (NODE_TYPE *head)
{
  NODE_TYPE *temp;

  temp = head;
  if (temp == NULL) {
    printf ("List is empty\n");
    return;
  }
  printf ("print (head): ");
  while (temp->next != NULL) {
    printf ("%d ", temp->data);
    temp = temp->next;
  }
  printf ("%d \n", temp->data);
}


```

```c
/* ------------------------------------------------------------ */
void dll_print_tail (NODE_TYPE *tail)
{
  NODE_TYPE *temp;

  temp = tail;
  if (temp == NULL) {
    printf ("List is empty\n");
    return;
  }
  printf ("print  (tail): ");
  while (temp->prev != NULL) {
    printf ("%d ", temp->data);
    temp = temp->prev;
  }
  printf ("%d \n", temp->data);
}


/* ------------------------------------------------------------ */
void dll_insert_head (NODE_TYPE **head, NODE_TYPE **tail, int data)
{
  NODE_TYPE *new_node;

  new_node = malloc (sizeof (NODE_TYPE));
  new_node->data = data;
  new_node->prev = NULL;
  new_node->next = NULL;

  if (*head == NULL) {
    *head = new_node;
    *tail = *head;
  }
  else {
    new_node->next = *head;
    (*head)->prev = new_node;
    *head = new_node;
  }
}


/* ------------------------------------------------------------ */
void dll_insert_tail (NODE_TYPE **head, NODE_TYPE **
    tail, int data)
{
  NODE_TYPE *new_node;

  new_node = malloc (sizeof (NODE_TYPE));
  new_node->data = data;
  new_node->prev = NULL;
  new_node->next = NULL;

  if (*head == NULL) {
    *head = new_node;
    new_node = *head;
  }
  else {
    (*tail)->next = new_node;
    new_node->prev = *tail;
    *tail = new_node;
  }
}


/* ------------------------------------------------------------ */
int main ()
{
  NODE_TYPE *head;
  NODE_TYPE *tail;
  int i, data;

  head = NULL;
  tail = NULL;
  for (i = 0; i < DATA_SIZE; i++) {
    data = (int) rand () % 100;
    printf ("adding (head): ");
    printf ("%02d\n", data);
    dll_insert_head (&head, &tail, data);
  }
  dll_print_head (head);
  dll_print_tail (tail);

  for (i = 0; i < DATA_SIZE; i++) {
    data = (int) rand () % 100;
    printf ("adding (tail): ");
    printf ("%02d\n", data);
    dll_insert_tail (&head, &tail, data);
  }
  dll_print_head (head);
  dll_print_tail (tail);

  return 0;
}
```

q14-2.c

```c
/* code: q14-2.c    (v1.20.00) */
#include<stdio.h>
#include<stdlib.h>

#define DATA_SIZE 6

/* circular doubly linked list */
struct node
{
  int data;
  struct node *prev;
  struct node *next;
};
typedef struct node NODE_TYPE;

/* ------------------------------------------------ */
void cdll_print_head (NODE_TYPE *head, NODE_TYPE *tail)
{

  if ((head == tail) && (head == NULL)) {
    printf ("List is empty\n");
    return;
  }
  printf ("print  (head): ");
  while (head->next != tail->next) {
    printf ("%d ", head->data);
    head = head->next;
  }
  printf ("%d \n", head->data);
}


/* ------------------------------------------------ */
void cdll_print_tail (NODE_TYPE *head, NODE_TYPE *tail)
{

  if ((head == tail) && (head == NULL)) {
    printf ("List is empty\n");
    return;
  }
  printf ("print (tail): ");
  while (head->prev != tail->prev) {
    printf ("%d ", tail->data);
    tail = tail->prev;
  }
  printf ("%d \n", tail->data);
}


/* ------------------------------------------------------------ */
void cdll_insert_head (NODE_TYPE **head, NODE_TYPE **
    tail, int data)
{
  NODE_TYPE *new_node;

  new_node = malloc (sizeof (NODE_TYPE));
  new_node->data = data;
  new_node->prev = NULL;
  new_node->next = NULL;

  if ((*head == *tail) && (*head == NULL)) {
    *head = new_node;
    *tail = new_node;
    (*head)->prev = NULL;
    (*head)->next = NULL;
    (*tail)->prev = NULL;
    (*tail)->next = NULL;
  }
  else {
    new_node->next = *head;
    (*head)->prev = new_node;
    *head = new_node;
    (*head)->prev = *tail;
```

```
      (*tail)->next = *head;
    }
}




/* -------------------------------------------------------- */
void cdll_insert_tail (NODE_TYPE **head, NODE_TYPE **
    tail, int data)
{
  NODE_TYPE *new_node;

  new_node = malloc (sizeof (NODE_TYPE));
  new_node->data = data;
  new_node->prev = NULL;
  new_node->next = NULL;

  if ((*head == *tail) && (*head == NULL)) {
    *head = new_node;
    *tail = new_node;
    (*head)->prev = NULL;
    (*head)->next = NULL;
    (*tail)->prev = NULL;
    (*tail)->next = NULL;
  }
  else {
    (*tail)->next = new_node;
    new_node->prev = *tail;
    *tail = new_node;
    (*head)->prev = *tail;
    (*tail)->next = *head;
  }
}


/* -------------------------------------------------------- */
int main ()
```

```
{
  NODE_TYPE *head;
  NODE_TYPE *tail;
  int i, data;

  head = NULL;
  tail = NULL;
  for (i = 0; i < DATA_SIZE; i++) {
    data = (int) rand () % 100;
    printf ("adding (head): ");
    printf ("%02d\n", data);
    cdll_insert_head (&head, &tail, data);
  }
  cdll_print_head (head, tail);
  cdll_print_tail (head, tail);

  for (i = 0; i < DATA_SIZE; i++) {
    data = (int) rand () % 100;
    printf ("adding (tail): ");
    printf ("%02d\n", data);
    cdll_insert_tail (&head, &tail, data);
  }
  cdll_print_head (head, tail);
  cdll_print_tail (head, tail);

  return 0;
}
```