

## ex1-1.c

```
1  /* code: ex1-1.c    (v1.20.00) */  
2  #include <stdio.h>  
3  
4  int main ()  
5  {  
6      printf ("The Open University of Japan\n");  
7  
8      return 0;  
9  }
```

## ex1-2.c

```
1  /* code: ex1-2.c    (v1.20.00) */
2  #include <stdio.h>
3
4  int main ()
5  {
6      int a;
7      printf ("Enter an integer: ");
8      scanf ("%d", &a);
9      printf ("The integer you entered was %d.\n", a);
10
11     return 0;
12 }
13
14 /* In Visual Studio (C++), it is recommended to
15 use the scanf_s() function instead of the scanf()
16 function used in Chapter 1.
17 However, it is possible to use scanf() with
18 the setting #pragma warning(disable:4996). */
```

## ex1-3.c

```
1  /* code: ex1-3.c    (v1.20.00) */
2  #include <stdio.h>
3
4  int main ()
5  {
6      char a;
7      short b;
8      int c;
9      long d;
10     float e;
11     double f;
12     printf ("char:--%zd-byte(s)\n", sizeof (a));
13     printf ("short:--%zd-byte(s)\n", sizeof (b));
14     printf ("int:---%zd-byte(s)\n", sizeof (c));
15     printf ("long:--%zd-byte(s)\n", sizeof (d));
16     printf ("float:--%zd-byte(s)\n", sizeof (e));
17     printf ("double:%zd-byte(s)\n", sizeof (f));
18
19     return 0;
20
21 }
```

## ex1-4.c

```
1  /* code: ex1-4.c    (v1.20.00) */
2  #include <stdio.h>
3
4  int main ()
5  {
6      int a, b, c;
7      a = 10;
8      b = 3;
9      c = 0;
10     printf ("a=%d\n", a);
11     printf ("b=%d\n\n", b);
12     c = a + b;
13     printf ("a+-b-=%d\n", c);
14     c = a - b;
15     printf ("a--b-=%d\n", c);
16     c = a * b;
17     printf ("a-*b-=%d\n", c);
18     c = a / b;
19     printf ("a-/b-=%d\n", c);
20     c = a % b;
21     printf ("a-%%b-=%d\n", c);
22
23     return 0;
24 }
```

## ex1-5.c

```
1  /* code: ex1-5.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <math.h>
4
5  int main ()
6  {
7      double x, y, z;
8      x = 30.0;
9      y = 3.0;
10     z = 0.0;
11     printf ("x=%f\n", x);
12     printf ("y=%f\n\n", y);
13     z = pow (x, y);
14     printf ("pow(x,y) = %f\n", z);
15
16     return 0;
17 }
```

## ex1-6.c

```
1  /* code: ex1-6.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <math.h>
4
5  int main ()
6  {
7      float celsius , fahrenheit;
8
9      celsius = 36.5;
10     fahrenheit = (9.0 / 5.0) * celsius + 32.0;
11     printf ("%f(Celsius) = %f(Fahrenheit)\n", celsius ,
12             fahrenheit);
13
14     return 0;
15 }
```

## ex1-7.c

```
1  /* code: ex1-7.c    (v1.20.00) */
2  #include <stdio.h>
3
4  int main ()
5  {
6      printf ("The Open University of Japan\n");
7      /* web address
8         http://www.ouj.ac.jp/    */
9
10     // C++ style comments
11     // C99 allows single-line comments
12
13     return 0;
14 }
```

## q1-1.c

```
1  /* code: q1-1.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <math.h>
4
5  int main ()
6  {
7      double x, y;
8
9      x = 3.14159;
10     y = 0.0;
11     printf ("x=%f\n\n", x);
12     y = ceil (x);
13     printf ("ceil(x) = %f\n", y);
14     y = floor (x);
15     printf ("floor(x) = %f\n", y);
16
17     return 0;
18 }
```



## q1-2.c

```
1  /* code: q1-2.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <math.h>
4
5  int main ()
6  {
7      float fx, fz;
8      double dx, dz;
9      long double lx, lz;
10
11     fx = 100.00F;
12     fz = sqrtf (fx);
13     printf ("fx = %f\n", fx);
14     printf ("sqrtf(fx) = %f\n\n", fz);
15
16     dx = 100.00;
17     dz = sqrt (dx);
18     printf ("dx = %f\n", dx);
19     printf ("sqrt(dx) = %f\n\n", dz);
20
21     lx = 100.00L;
22     lz = sqrtl (lx);
23     printf ("lx = %Lf\n", lx);
24     printf ("sqrtl(lx) = %Lf\n\n", lz);
25
26     return 0;
27 }
```

## q1-3.c

```
1  /* code: q1-3.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <math.h>
4
5  int main ()
6  {
7      float fahrenheit, celsius;
8
9      fahrenheit = 25.1;
10     celsius = (5.0 / 9.0) * (fahrenheit - 32.0);
11     printf ("%f(Fahrenheit) = %f(Celsius)\n", fahrenheit,
12             celsius);
13
14     return 0;
15 }
```

## ex2-1.c

```
1  /* code: ex2-1.c    (v1.20.00) */
2  #include <stdio.h>
3
4  int main ()
5  {
6      int x, y;
7
8      x = 500;
9      y = 300;
10
11     printf ("X=%d\n", x);
12     printf ("Y=%d\n", y);
13
14     if (x > y)
15         printf ("X is greater than Y.\n");
16
17
18     return 0;
19 }
```

## ex2-2.c

```
1  /* code: ex2-2.c    (v1.20.00) */
2  #include <stdio.h>
3
4  int main ()
5  {
6      int x, y;
7      x = 500;
8      y = 700;
9      printf ("X=%d\n", x);
10     printf ("Y=%d\n", y);
11
12     if (x > y)
13         printf ("X is greater than Y.\n");
14     else
15         printf ("X is less than or equal to Y.\n");
16
17     return 0;
18 }
```

## ex2-3.c

```
1  /* code: ex2-3.c    (v1.20.00) */
2  #include <stdio.h>
3
4  int main ()
5  {
6      char grade;
7
8      grade = 'B';
9
10     switch (grade) {
11     case 'A':
12         printf ("excellent\n");
13         break;
14     case 'B':
15         printf ("good\n");
16         break;
17     case 'C':
18         printf ("fair\n");
19         break;
20     case 'D':
21         printf ("barely - passing\n");
22         break;
23     case 'F':
24         printf ("not - passing\n");
25         break;
26     default:
27         printf ("ERROR: - invalid - character\n");
28         break;
29     }
30     printf ("Your - grade - is - %c\n", grade);
31     return 0;
32 }
```

## q2-1.c

```
1  /* code: q2-1.c    (v1.20.00) */
2  #include <stdio.h>
3
4  int main ()
5  {
6      int x, y;
7
8      printf ("enter X: ");
9      scanf ("%d", &x);
10     printf ("enter Y: ");
11     scanf ("%d", &y);
12
13     printf ("X=%d\n", x);
14     printf ("Y=%d\n", y);
15
16     if (x > y) {
17         printf ("X is greater than Y.\n");
18     }
19     else {
20         printf ("X is less than or equal to Y.\n");
21     }
22
23     return 0;
24 }
```

## q2-2.c

```
1  /* code: q2-2.c    (v1.20.00) */
2  #include <stdio.h>
3
4  int main ()
5  {
6      char grade;
7
8      grade = 'b';
9
10     switch (grade) {
11     case 'a':
12     case 'A':
13         printf ("excellent\n");
14         break;
15     case 'b':
16     case 'B':
17         printf ("good\n");
18         break;
19     case 'c':
20     case 'C':
21         printf ("fair\n");
22         break;
23     case 'd':
24     case 'D':
25         printf ("barely - passing\n");
26         break;
27     case 'f':
28     case 'F':
29         printf ("not - passing\n");
30         break;
31     default:
32         printf ("ERROR: - invalid - character\n");
33         break;
34     }
35     printf ("Your - grade - is - %c\n", grade);
```

```
36     return 0;  
37 }
```



## q2-3.c

```
1  /* code: q2-3.c    (v1.20.00) */
2  #include <stdio.h>
3
4  int main ()
5  {
6      int a;
7      a = 3;
8
9      if (a == 0 || a == 1 || a == 2) {
10         printf ("A\n");
11     }
12     else if (a == 3 || a == 4) {
13         printf ("B\n");
14     }
15     else {
16         printf ("ERROR: -invalid -number\n");
17     }
18
19     return 0;
20 }
```

## q2-3a.c

```
1  /* code: q2-3a.c    (v1.20.00) */
2  #include <stdio.h>
3
4  int main ()
5  {
6      int a;
7      a = 3;
8      switch (a) {
9          case 0:
10         case 1:
11         case 2:
12             printf ("A\n");
13             break;
14         case 3:
15         case 4:
16             printf ("B\n");
17             break;
18         default:
19             printf ("ERROR: - invalid - number\n");
20             break;
21     }
22
23     return 0;
24 }
```

## ex3-1.c

```
1  /* code: ex3-1.c    (v1.20.00) */
2  #include <stdio.h>
3
4  int main ()
5  {
6      int i;
7
8      for (i = 0; i < 10; i++)
9          printf ("%d", i);
10
11     return 0;
12 }
```

## ex3-10.c

```
1  /* code: ex3-10.c    (v1.20.00) */
2  #include <stdio.h>
3
4  int main ()
5  {
6      int i;
7
8      i = 0;
9      while (1) {
10         printf ("%d", i);
11         if (i == 5) {
12             i = 0;
13             break;
14         }
15         i++;
16     }
17
18     return 0;
19 }
```

## ex3-11.c

```
1  /* code: ex3-11.c    (v1.20.00) */
2  #include <stdio.h>
3
4  int main ()
5  {
6      int i;
7
8      i = 0;
9      while (1) {
10         printf ("%d", i);
11         if (i == 5) {
12             i = 0;
13             continue;
14         }
15         i++;
16     }
17
18     return 0;
19 }
```

## ex3-2.c

```
1  /* code: ex3-2.c    (v1.20.00) */
2  #include <stdio.h>
3
4  int main ()
5  {
6      int i;
7
8      for (i = 0; i < 10; i++) {
9          printf ("%d", i);
10         if (0 != (i % 2))
11             printf (":odd-");
12         else
13             printf (":even-");
14     }
15
16     return 0;
17 }
```

## ex3-3.c

```
1  /* code: ex3-3.c    (v1.20.00) */
2  #include <stdio.h>
3
4  int main ()
5  {
6      int i, j;
7
8      for (i = 1; i < 10; i++) {
9          for (j = 1; j < 10; j++) {
10             printf ("%02d-", i * j);
11         }
12         printf ("\n");
13     }
14
15     return 0;
16 }
```

## ex3-4.c

```
1  /* code: ex3-4.c    (v1.20.00) */
2  #include <stdio.h>
3
4  int main ()
5  {
6      int i;
7
8      i = 0;
9      for (;;) {
10         printf ("%d", i);
11         i++;
12     }
13     return 0;
14 }
```



## ex3-5.c

```
1  /* code: ex3-5.c    (v1.20.00) */
2  #include <stdio.h>
3
4  int main ()
5  {
6      int i;
7
8      i = 0;
9      while (i < 10)
10         printf ("%d ", i++);
11
12     return 0;
13 }
```

## ex3-6.c

```
1  /* code: ex3-6.c    (v1.20.00) */
2  #include <stdio.h>
3
4  int main ()
5  {
6      int i;
7
8      i = 0;
9      while (i < 10) {
10         printf ("%d", i);
11         i++;
12     }
13     return 0;
14 }
```

## ex3-7.c

```
1  /* code: ex3-7.c    (v1.20.00) */
2  #include <stdio.h>
3
4  int main ()
5  {
6      int i;
7
8      i = 0;
9      do {
10         printf ("%d", i);
11         i++;
12     } while (i < 10);
13
14     return 0;
15 }
```

## ex3-8.c

```
1  /* code: ex3-8.c    (v1.20.00) */
2  #include <stdio.h>
3
4  int main ()
5  {
6      int i, j;
7
8      i = 1;
9      while (i < 10) {
10         j = 1;
11         while (j < 10) {
12             printf ("%02d-", i * j);
13             j++;
14         }
15         printf ("\n");
16         i++;
17     }
18
19     return 0;
20 }
```

## ex3-9.c

```
1  /* code: ex3-9.c    (v1.20.00) */
2  #include <stdio.h>
3
4  int main ()
5  {
6      int i;
7
8      i = 0;
9      while (1) {
10         printf ("%d ", i);
11         i++;
12     }
13
14     return 0;
15 }
```

## q3-1.c

```
1  /* code: q3-1.c    (v1.20.00) */
2  #include <stdio.h>
3
4  int main ()
5  {
6      int i;
7
8      for (i = 0; i < 100; i++)
9          printf ("%d", i);
10
11     return 0;
12 }
```

## q3-2.c

```
1  /* code: q3-2.c    (v1.20.00) */
2  #include <stdio.h>
3
4  int main ()
5  {
6      int i;
7
8      for (i = 9; i >= 0; i--)
9          printf ("%d ", i);
10
11     return 0;
12 }
```

## q3-3.c

```
1  /* code: q3-3.c    (v1.20.00) */
2  #include <stdio.h>
3
4  int main ()
5  {
6      int i;
7
8      i = 0;
9      while (i < 100) {
10         printf ("%d", i);
11         i++;
12     }
13
14     return 0;
15 }
```



## q3-4.c

```
1  /* code: q3-4.c    (v1.20.00) */
2  #include <stdio.h>
3
4  int main ()
5  {
6      int i, j, k;
7
8      for (i = 0; i < 2; i++) {
9          for (j = 0; j < 2; j++) {
10             for (k = 0; k < 2; k++) {
11                 printf ("%d-%d-%d", i, j, k);
12                 printf ("\n");
13             }
14         }
15     }
16
17     return 0;
18 }
```

## q3-5.c

```
1  /* code: q3-5.c    (v1.20.00) */
2  #include <stdio.h>
3
4  int main ()
5  {
6      int i, j, k;
7
8      for (i = 0; i < 2; i++) {
9          for (j = 0; j < 2; j++) {
10             for (k = 0; k < 2; k++) {
11                 printf ("%d-", i * j + k);
12             }
13         }
14     }
15
16     return 0;
17 }
```

## ex4-1.c

```
1  /* code: ex4-1.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  #define POINTS 1000
6
7  int main ()
8  {
9      int i, count, points;
10     double x, y, q;
11     double pi;
12
13     points = POINTS;
14     count = 0;
15
16     for (i = 0; i < points; i++) {
17         x = (double) rand () / ((double) RAND_MAX + 1.0);
18         y = (double) rand () / ((double) RAND_MAX + 1.0);
19         q = (x * x) + (y * y);
20
21         if (q <= 1.00)
22             count++;
23     }
24
25     pi = (double) count / (double) points *(double) 4.00;
26     printf ("circle: %d\t", count);
27     printf ("square: %d\t", points);
28     printf ("PI: %f\n", pi);
29
30     return 0;
31 }
```

## ex4-2.c

```
1  /* code: ex4-2.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <math.h>
5
6  int main ()
7  {
8      int i, j, count, points;
9      double x, y, q;
10     double pi;
11
12     for (j = 1; j < 10; j++) {
13         points = 1;
14         count = 0;
15         points = points * pow (10, j);
16         for (i = 0; i < points; i++) {
17             x = (double) rand () / ((double) RANDMAX + 1.0);
18             y = (double) rand () / ((double) RANDMAX + 1.0);
19             q = (x * x) + (y * y);
20
21             if (q <= 1.00)
22                 count++;
23         }
24         pi = (double) count / (double) points *(double)
25             4.00;
26         printf ("circle: -%10d\t", count);
27         printf ("square: -%10d\t", points);
28         printf ("PI: -%f -(%+f)\n", pi, (pi - M_PI));
29     }
30     return 0;
}
```

## ex5-1.c

```
1  /* code: ex5-1.c    (v1.20.00) */
2  #include <stdio.h>
3
4  int main ()
5  {
6      int i;
7
8      for (i = 0; i < 10; i++)
9          printf ("%d-", i);
10     printf ("\n");
11
12     for (i = 0; i < 10; i++)
13         printf ("%d-", i);
14     printf ("\n");
15
16     for (i = 0; i < 10; i++)
17         printf ("%d-", i);
18     printf ("\n");
19
20     return 0;
21 }
```

## ex5-2.c

```
1  /* code: ex5-2.c    (v1.20.00) */
2  #include <stdio.h>
3
4  void print_numbers (void)
5  {
6      int i;
7
8      for (i = 0; i < 10; i++)
9          printf ("%d-", i);
10     printf ("\n");
11 }
12
13 int main ()
14 {
15     print_numbers ();
16     print_numbers ();
17     print_numbers ();
18     return 0;
19 }
```

## ex5-3.c

```
1  /* code: ex5-3.c    (v1.20.00) */
2  #include <stdio.h>
3
4  void print_numbers (void);
5
6  int main ()
7  {
8      print_numbers ();
9      print_numbers ();
10     print_numbers ();
11     return 0;
12 }
13
14 void print_numbers (void)
15 {
16     int i;
17
18     for (i = 0; i < 10; i++)
19         printf ("%d-", i);
20     printf ("\n");
21 }
```

## ex5-4.c

```
1  /* code: ex5-4.c    (v1.20.00) */
2  #include <stdio.h>
3
4  void g (void)
5  {
6      int i;
7      for (i = 0; i < 3; i++) {
8          printf ("a");
9      }
10 }
11
12 void f (void)
13 {
14     int i;
15     for (i = 0; i < 5; i++) {
16         g ();
17     }
18 }
19
20 int main (int argc, char **argv)
21 {
22     f ();
23     return 0;
24 }
```



## ex5-5.c

```
1  /* code: ex5-5.c    (v1.20.00) */
2  #include <stdio.h>
3
4  float triangle (float base, float height)
5  {
6      float c;
7      c = (base * height) / 2.000F;
8      return c;
9  }
10
11 int main ()
12 {
13     float t;
14     t = triangle (3.00, 4.00);
15     printf ("triangle = %f\n", t);
16     t = triangle (5.00, 6.00);
17     printf ("triangle = %f\n", t);
18
19     return 0;
20 }
```

## ex5-6.c

```
1  /* code: ex5-6.c    (v1.20.00) */
2  #include <stdio.h>
3
4  void add_pass_by_value (int i)
5  {
6      i = i + 1;
7  }
8
9  void add_pass_by_reference (int *i)
10 {
11     *i = *i + 1;
12 }
13
14 int main ()
15 {
16     int a;
17
18     a = 10;
19     add_pass_by_value (a);
20     printf ("%d\n", a);
21
22     a = 10;
23     add_pass_by_reference (&a);
24     printf ("%d\n", a);
25
26     return 0;
27 }
```

## ex5-7.c

```
1  /* code: ex5-7.c    (v1.20.00) */
2  #include <stdio.h>
3
4  int factorial (int n)
5  {
6      if (n == 0) {
7          return 1;
8      }
9      else {
10         return n * factorial (n - 1);
11     }
12 }
13
14 int main ()
15 {
16     int i;
17     i = 5;
18     printf ("%d! = %d\n", i, factorial (i));
19
20     return 0;
21 }
```

## q5-1.c

```
1  /* code: q5-1.c    (v1.20.00) */
2  #include <stdio.h>
3
4  float trapezoid (float a, float b, float h)
5  {
6      float c;
7      c = ((a + b) / 2.000F) * h;
8      return c;
9  }
10
11 int main ()
12 {
13     float t;
14     t = trapezoid (3.00, 4.00, 5.00);
15     printf ("trapezoid = %f\n", t);
16     t = trapezoid (5.00, 6.00, 7.00);
17     printf ("trapezoid = %f\n", t);
18
19     return 0;
20 }
```

## q5-2.c

```
1  /* code: q5-2.c    (v1.20.00) */
2  #include <stdio.h>
3
4  struct student
5  {
6      int id;
7      char grade;
8      float average;
9  };
10 typedef struct student STUDENT_TYPE;
11
12 STUDENT_TYPE initialize_student_record (STUDENT_TYPE s)
13 {
14     s.id++;
15     s.grade = 'x';
16     s.average = 0.0;
17     return s;
18 }
19
20 int main ()
21 {
22     STUDENT_TYPE student;
23
24     student.id = 20;
25     student.grade = 'a';
26     student.average = 300.000;
27     printf ("%d-%c-%f\n", student.id, student.grade,
28             student.average);
29     student = initialize_student_record (student);
30     printf ("%d-%c-%f\n", student.id, student.grade,
31             student.average);
32
33     return 0;
34 }
```

## q5-3.c

```
1  /* code: q5-3.c    (v1.20.00) */
2  #include <stdio.h>
3
4  int fibonacci (int n)
5  {
6      if (n == 0) {
7          return 0;
8      }
9      else if (n == 1) {
10         return 1;
11     }
12     else {
13         return (fibonacci (n - 1) + fibonacci (n - 2));
14     }
15 }
16
17 int main ()
18 {
19     int i;
20     i = 10;
21     printf ("fibonacci(%d) = %d\n", i, fibonacci (i));
22
23     return 0;
24 }
```

## q5-4.c

```
1  /* code: q5-4.c    (v1.20.00) */
2  #include <stdio.h>
3
4  void foo (int n)
5  {
6      if (n < 15) {
7          foo (n + 1);
8          printf ("%d-", n);
9      }
10 }
11
12 int main ()
13 {
14
15     foo (0);
16
17     return 0;
18 }
```

## q5-5.c

```
1  /* code: q5-5.c    (v1.20.00) */
2  #include <stdio.h>
3
4  int factorial (int n)
5  {
6      if (n == 0) {
7          return 1;
8      }
9      else {
10         return n * factorial (n - 1);
11     }
12 }
13
14 int main ()
15 {
16     int i;
17     /* i = -1; */
18     i = 1;
19     printf ("%d! = %d\n", i, factorial (i));
20
21     return 0;
22 }
```



## ex6-1.c

```
1  /* code: ex6-1.c    (v1.20.00) */
2  #include <stdio.h>
3
4  int main ()
5  {
6      int a, b, c, d, e;
7      int sum, avg;
8
9      a = 30;
10     b = 20;
11     c = 10;
12     d = 25;
13     e = 15;
14     sum = a + b + c + d + e;
15     avg = sum / 5;
16     printf ("%d\n", avg);
17
18     return 0;
19 }
```

## ex6-10.c

```
1  /* code: ex6-10.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <string.h>
4
5  int main ()
6  {
7      char s0 [] = "aaaaa";
8      char s1 [] = "bbbbb";
9      char s2 [] = "aaaaaaa";
10     int i;
11     printf ("strcmp(str1 , -str2)\n");
12     i = strcmp (s0, s0);
13     printf ("%s] - [%s] - (%d)\n", s0, s0, i);
14     i = strcmp (s0, s1);
15     printf ("%s] - [%s] - (%d)\n", s0, s1, i);
16     i = strcmp (s1, s0);
17     printf ("%s] - [%s] - (%d)\n", s1, s0, i);
18     i = strcmp (s0, s2);
19     printf ("%s] - [%s] - (%d)\n", s0, s2, i);
20
21     return 0;
22 }
```

## ex6-2.c

```
1  /* code: ex6-2.c    (v1.20.00) */
2  #include <stdio.h>
3
4  int main ()
5  {
6      int a[10];
7      int i, sum, avg;
8
9      a[0] = 30;
10     a[1] = 20;
11     a[2] = 10;
12     a[3] = 25;
13     a[4] = 15;
14     sum = 0;
15     for (i = 0; i < 5; i++)
16         sum += a[i];
17
18     avg = sum / 5;
19     printf ("%d\n", avg);
20
21     return 0;
22 }
```

## ex6-3.c

```
1  /* code: ex6-3.c    (v1.20.00) */
2  #include <stdio.h>
3
4  int main ()
5  {
6      int a[10] = { 30, 20, 10, 25, 15 };
7      int i, sum, avg;
8
9      sum = 0;
10     for (i = 0; i < 5; i++)
11         sum += a[i];
12
13     avg = sum / 5;
14     printf ("%d\n", avg);
15
16     return 0;
17 }
```

## ex6-4.c

```
1  /* code: ex6-4.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  #define ARRAY_SIZE 10
6
7  int main ()
8  {
9      int a[ARRAY_SIZE];
10     int i;
11
12     for (i = 0; i < ARRAY_SIZE; i++)
13         a[i] = rand () % 100;
14
15     for (i = 0; i < ARRAY_SIZE; i++)
16         printf ("%03d-", a[i]);
17
18     return 0;
19 }
```

## ex6-5.c

```
1  /* code: ex6-5.c    (v1.20.00) */
2  #include <stdio.h>
3
4  int main ()
5  {
6      int i, j;
7      int a[3][4] = {
8          {0, 10, 20, 30},
9          {40, 50, 60, 70},
10         {80, 90, 100, 110}
11     };
12
13     for (i = 0; i < 3; i++) {
14         for (j = 0; j < 4; j++) {
15             printf ("array[%d][%d]=%3d\n", i, j, a[i][j]);
16         }
17     }
18
19     return 0;
20 }
```

## ex6-6.c

```
1  /* code: ex6-6.c    (v1.20.00) */
2  #include <stdio.h>
3
4  int main ()
5  {
6      int i, j, k;
7      int a[2][3][4] = {
8          {0, 1, 2, 3},
9          {4, 5, 6, 7},
10         {8, 9, 10, 11}},
11         {{0, 10, 20, 30},
12          {40, 50, 60, 70},
13          {80, 90, 100, 110}}
14     };
15
16     for (i = 0; i < 2; i++) {
17         for (j = 0; j < 3; j++) {
18             for (k = 0; k < 4; k++) {
19                 printf ("array[%d][%d][%d]=%3d\n", i, j, k, a[i
20                     ][j][k]);
21             }
22         }
23     }
24     return 0;
25 }
```

## ex6-7.c

```
1  /* code: ex6-7.c    (v1.20.00) */
2  #include <stdio.h>
3
4  int main ()
5  {
6      char s[4];
7      s[0] = 'O';
8      s[1] = 'U';
9      s[2] = 'J';
10     s[3] = '\0';
11     printf ("%s\n", s);
12
13     return 0;
14 }
```



## ex6-8.c

```
1  /* code: ex6-8.c    (v1.20.00) */
2  #include <stdio.h>
3
4  /* ----- */
5  void string_copy (char *target, char *source)
6  {
7      int i;
8      i = 0;
9      while (source[i] != '\0') {
10         target[i] = source[i];
11         i++;
12     }
13     target[i] = '\0';
14 }
15
16 /* ----- */
17 int main ()
18 {
19     char s[20] = "University";
20     char t[20];
21
22     string_copy (t, s);
23     printf ("%s\n", t);
24
25     return 0;
26 }
```

## ex6-9.c

```
1  /* code: ex6-9.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <string.h>
4
5  /* _____ */
6  int main ()
7  {
8      char s[20] = "University";
9      char t[20];
10     strcpy (t, s);
11     printf ("%s\n", t);
12
13     return 0;
14 }
```

## q6-1.c

```
1  /* code: q6-1.c    (v1.20.00) */
2  #include <stdio.h>
3
4  int main ()
5  {
6      float a[5];
7      int i;
8      float sum, avg;
9
10     a[0] = 30.0;
11     a[1] = 20.0;
12     a[2] = 10.0;
13     a[3] = 25.0;
14     a[4] = 15.0;
15     sum = 0.0;
16     for (i = 0; i < 5; i++)
17         sum += a[i];
18
19     avg = sum / 5.00;
20     printf ("%f\n", avg);
21
22     return 0;
23 }
```

## q6-2.c

```
1  /* code: q6-2.c    (v1.20.00) */
2  #include <stdio.h>
3  #define TABLE 9
4  int main ()
5  {
6      int i, j;
7      int a[TABLE][TABLE];
8
9      for (i = 0; i < TABLE; i++) {
10         for (j = 0; j < TABLE; j++) {
11             a[i][j] = (i + 1) * (j + 1);
12         }
13     }
14     for (i = 0; i < TABLE; i++) {
15         for (j = 0; j < TABLE; j++) {
16             printf ("%02d-", a[i][j]);
17         }
18         printf ("\n");
19     }
20     return 0;
21 }
```

## q6-3.c

```
1  /* code: q6-3.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  int main ()
6  {
7      int i, j, k;
8      int array[2][3][4];
9
10     for (i = 0; i < 2; i++) {
11         for (j = 0; j < 3; j++) {
12             for (k = 0; k < 4; k++) {
13                 array[i][j][k] = (rand () % 100) + 1;
14             }
15         }
16     }
17     for (i = 0; i < 2; i++) {
18         for (j = 0; j < 3; j++) {
19             for (k = 0; k < 4; k++) {
20                 printf ("%03d ", array[i][j][k]);
21             }
22             printf ("\n");
23         }
24         printf ("\n");
25     }
26     return 0;
27 }
```

## q6-4.c

```
1  /* code: q6-4.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <string.h>
4
5  /* _____ */
6  int main ()
7  {
8      char s0 [] = "aaaaa";
9      char s1 [] = "bbbbb";
10     char s2 [] = "aaaaaaa";
11     int i;
12     printf ("strncmp(str1 , -str2 , -3)\n");
13     i = strncmp (s0 , s0 , 3);
14     printf ("%s] - [%s] - (%d)\n" , s0 , s0 , i);
15     i = strncmp (s0 , s1 , 3);
16     printf ("%s] - [%s] - (%d)\n" , s0 , s1 , i);
17     i = strncmp (s1 , s0 , 3);
18     printf ("%s] - [%s] - (%d)\n" , s1 , s0 , i);
19     i = strncmp (s0 , s2 , 3);
20     printf ("%s] - [%s] - (%d)\n" , s0 , s2 , i);
21
22     return 0;
23 }
```

## q6-5.c

```
1  /* code: q6-5.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <string.h>
4
5  int main ()
6  {
7      char s0 [] = "ABCDEFGH";
8      int i;
9      i = strlen (s0);
10     printf ("%s (%d)\n", s0, i);
11     return 0;
12 }
```

## q6-6.c

```

1  /* code: q6-6.c    (v1.20.00) */
2
3  /* In Visual Studio (C++), the error (E0513, C2440)
4  in q6-6.c can be avoided by casting.
5  Using the strcpy_s() function is also recommended. */
6
7  #include <stdio.h>
8  #include <stdlib.h>
9  #include <string.h>
10
11 #define MAX 10
12
13 struct student
14 {
15     int id;
16     char grade;
17     char name[128];
18 };
19 typedef struct student STUDENT_TYPE;
20
21 /* ----- */
22 int main ()
23 {
24     STUDENT_TYPE db1[MAX];
25     STUDENT_TYPE *db2[MAX];
26     int i;
27
28     printf ("database1\n");
29     for (i = 0; i < MAX; i++) {
30         db1[i].id = 100 + i;
31         db1[i].grade = 'a' + rand () % 5;
32         strcpy (db1[i].name, "John-Doe");
33         printf ("%d-%c-%s\n", db1[i].id, db1[i].grade, db1[
34             i].name);
35     }

```



```
35
36     printf ("\n");
37     printf ("database2\n");
38     for (i = 0; i < MAX; i++) {
39         /* db2[i] = (STUDENT_TYPE*) malloc(sizeof(
           STUDENT_TYPE)); */
40         db2[i] = malloc (sizeof (STUDENT_TYPE));
41         db2[i]—>id = 200 + i;
42         db2[i]—>grade = 'a' + rand () % 5;
43         strcpy (db2[i]—>name, "John·Doe");
44         printf ("%d·%c·%s\t\t", db2[i]—>id, db2[i]—>grade,
           db2[i]—>name);
45         printf ("%d·%c·%s\n", (*db2[i]).id, (*db2[i]).grade
           , (*db2[i]).name);
46     }
47     for (i = 0; i < MAX; i++) {
48         free (db2[i]);
49     }
50
51
52     return 0;
53 }
```

## ex7-1.c

```
1  /* code: ex7-1.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <stdlib.h>
4  #define ARRAY_SIZE 13
5
6  /* ----- */
7  int linear_search (int array[], int n, int key)
8  {
9      int i;
10     for (i = 0; i < n; i++) {
11         if (array[i] == key) {
12             return i;
13         }
14     }
15     return -1;
16 }
17
18 /* ----- */
19 void print_array (int array[], int n)
20 {
21     int i;
22     for (i = 0; i < n; i++) {
23         printf ("%d-", array[i]);
24     }
25     printf ("\n");
26 }
27
28 /* ----- */
29 int main ()
30 {
31     int index, key;
32     int array[ARRAY_SIZE] = {
33         900, 990, 210, 50, 80, 150, 330,
34         470, 510, 530, 800, 250, 280
35     };
```

```
36 | key = 800;
37 | print_array (array , ARRAY_SIZE);
38 | index = linear_search (array , ARRAY_SIZE, key);
39 | if (index != -1) {
40 |     printf ("Found: -%-d- (Index:%d)\n" , key , index);
41 | }
42 | else {
43 |     printf ("Not -found: -%-d\n" , key);
44 | }
45 | return 0;
46 | }
```

## ex7-2.c

```
1  /* code: ex7-2.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <stdlib.h>
4  #define ARRAY_SIZE 13
5
6  /* ----- */
7  int binary_search (int array[], int num, int key)
8  {
9      int middle, low, high;
10     low = 0;
11     high = num - 1;
12     while (low <= high) {
13         middle = (low + high) / 2;
14         if (key == array[middle]) {
15             return middle;
16         }
17         else if (key < array[middle]) {
18             high = middle - 1;
19         }
20         else {
21             low = middle + 1;
22         }
23     }
24     return -1;
25 }
26
27 /* ----- */
28 void print_array (int array[], int n)
29 {
30     int i;
31     for (i = 0; i < n; i++) {
32         printf ("%d ", array[i]);
33     }
34     printf ("\n");
35 }
```

```
36
37 /* ----- */
38 int main ()
39 {
40     int index , key;
41     int array[ARRAY_SIZE] = {
42         50, 80, 150, 210, 250, 280, 330,
43         470, 510, 530, 800, 900, 990
44     };
45
46     key = 800;
47     print_array (array , ARRAY_SIZE);
48     index = binary_search (array , ARRAY_SIZE, key);
49     if (index != -1) {
50         printf ("Found: %d (Index:%d)\n" , key , index);
51     }
52     else {
53         printf ("Not found: %d\n" , key);
54     }
55     return 0;
56 }
```

## q7-1.c

```
1  /* code: q7-1.c    (v1.25.00) */
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <search.h>
5
6  #define ARRAY_SIZE 5
7  #define EXTRAROOM 1
8
9  /* ----- */
10 int compare (const void *x, const void *y)
11 {
12
13     int a = *(const int *) x;
14
15     int b = *(const int *) y;
16
17     return (a > b) - (a < b);
18 }
19
20
21
22
23 /* ----- */
24 void print_array (int arr [], size_t n)
25 {
26
27     for (size_t i = 0; i < n; i++) {
28
29         printf ("%d-", arr[i]);
30
31     }
32     printf ("\n");
33
34 }
35
```

```
36
37 /* ----- */
38 int main ()
39 {
40
41     int key = 25;
42
43     size_t elements = ARRAY_SIZE;
44
45     int array[ARRAY_SIZE + EXTRA_ROOM] = { 10, 20, 30, 40,
46         50 };
47
48     printf ("Initial array:-");
49
50     print_array (array, elements);
51
52
53     int *found = lsearch (&key, array, &elements, sizeof (
54         int), compare);
55
56     if (!found) {
57
58         printf ("Search error\n");
59
60         return EXIT_FAILURE;
61     }
62
63
64     printf ("\nResult:\n");
65
66     if (elements == ARRAY_SIZE) {
67
68         printf ("Key-%d-was-already-in-the-array.\n", key);
69
70     }
71 }
```

```
72     else {  
73  
74     printf ( "Key %d was added to the array.\n" , key );  
75  
76     }  
77  
78  
79     print_array ( array , elements );  
80  
81     return EXIT_SUCCESS;  
82  
83 }
```



## q7-2.c

```
1  /* code: q7-2.c    (v1.25.00) */
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <search.h>
5
6  #define ARRAY_SIZE 10
7
8  /* ----- */
9  int compare (const void *a, const void *b)
10 {
11     int x = *(int *) a;
12     int y = *(int *) b;
13     if (x < y)
14         return -1;
15     if (x > y)
16         return 1;
17     return 0;
18 }
19
20 /* ----- */
21 void print_array (int array[], int n)
22 {
23     int i;
24     for (i = 0; i < n; i++) {
25         printf ("%d-", array[i]);
26     }
27     printf ("\n");
28 }
29
30 /* ----- */
31 int main ()
32 {
33     int key;
34     int *r;
35     int array[ARRAY_SIZE] = {
```

```
36     10, 12, 16, 19, 28, 30, 38, 44, 70, 98
37 };                                     /* ordered array! */
38
39 key = 16;
40 print_array (array , ARRAY_SIZE);
41
42 r = (int *) bsearch (&key, array , ARRAY_SIZE, sizeof
43     (int), compare);
44 if (r != NULL) {
45     printf ("Found: %d\n" , *r);
46 }
47 else {
48     printf ("Not found: %d\n" , key);
49 }
50 return 0;
51 }
```

## q7-3.c

```

1  /* code: q7-3.c    (v1.25.00) */
2
3  #include <stdio.h>
4  #include <stdlib.h>
5  #define MAX 1000000
6
7  /* ----- */
8  void array_print (int a[], int max)
9  {
10     int i;
11     for (i = 0; i < max; i++) {
12         printf ("%02d-", a[i]);
13     }
14     printf ("\n");
15 }
16
17 /* ----- */
18 int array_find_empty (int a[], int max)
19 {
20     int i;
21     for (i = 0; i < max; i++) {
22         if (a[i] == -1) {
23             return i;
24         }
25     }
26     return -1;
27 }
28
29 /* ----- */
30 void array_insert (int a[], int max, int index, int
    empty, int data)
31 {
32     int i;
33     if (empty > index) {
34         for (i = empty; i > index; i--) {

```

```

35     a[i] = a[i - 1];
36 }
37 }
38 else {
39     for (i = empty; i < index; i++) {
40         a[i] = a[i + 1];
41     }
42 }
43 a[index] = data;
44 }
45
46 /* ----- */
47 int array_delete (int a[], int index)
48 {
49     int data;
50     data = a[index];
51     a[index] = -1;
52     return data;
53 }
54
55
56 /* ----- */
57 int main ()
58 {
59     int i, j, index_ins, index_del, empty, data;
60
61     int *a;
62     a = (int *) malloc (sizeof (int) * MAX);
63     if (a == NULL) {
64         printf ("Error! memory not allocated.");
65         exit (EXIT_FAILURE);
66     }
67
68     for (j = 0; j < MAX; j++) {
69         a[j] = rand () % 100;
70     }
71     data = a[MAX / 2];
72     a[MAX / 2] = -1;

```

```

73
74     for (i = 0; i < 1000; i++) {
75         index_ins = rand () % MAX;
76         index_del = rand () % MAX;
77         /* printf("ins:%d  del:%d\n", index_ins , index_del
           ); */
78
79         empty = array_find_empty (a, MAX);
80         /* printf("empty:%d\n", empty ); */
81
82         array_insert (a, MAX, index_ins , empty, data);
83
84         data = array_delete (a, index_del);
85         /* array_print( a, MAX ); */
86     }
87
88
89     free (a);
90
91     return 0;
92 }
93
94 /* ----- */
95 /*
96 $ gcc -pg -Wall q7-3.c -o q7-3
97 $ ./q7-3
98 $ gprof q7-3 gmon.out > gmon.log
99 $ more gmon.log
100 */
101 /* ----- */

```

## ex8-1.c

```

1  /* code: ex8-1.c    (v1.20.00) */
2  #include<stdio.h>
3  #include<stdlib.h>
4
5  #define MAX 128
6  #define PUSH_SUCCESS    1
7  #define PUSH_FAILURE    -1
8  #define POP_SUCCESS     2
9  #define POP_FAILURE     -2
10
11 /* ----- */
12 void stack_init (int *top)
13 {
14     *top = 0;
15 }
16
17 /* ----- */
18 void display (int stack[], int top)
19 {
20     int i;
21     printf ("STACK(%d):-", top);
22     for (i = 0; i < top; i++) {
23         printf ("%d-", stack[i]);
24     }
25     printf ("\n");
26 }
27
28 /* ----- */
29 int push (int stack[], int *top, int data)
30 {
31     if (*top >= MAX) {
32         /* stack overflow */
33         return PUSH_FAILURE;
34     }
35     else {

```

```

36     stack[*top] = data;
37     (*top)++;
38     return PUSH_SUCCESS;
39 }
40 }
41
42 /* ----- */
43 int pop (int stack [], int *top, int *data)
44 {
45     if ((*top) > 0) {
46         *data = stack[*top - 1];
47         (*top)--;
48         return POP_SUCCESS;
49     }
50     else {
51         /* stack empty */
52         return POP_FAILURE;
53     }
54 }
55
56 /* ----- */
57 int main ()
58 {
59     int stack[MAX];
60     int top, data;
61
62     stack_init (&top);
63     data = 300;
64     printf ("push: %d\n", data);
65     push (stack, &top, data);
66     data = 400;
67     printf ("push: %d\n", data);
68     push (stack, &top, data);
69     data = 500;
70     printf ("push: %d\n", data);
71     push (stack, &top, data);
72     display (stack, top);
73     pop (stack, &top, &data);

```

```
74     printf ("pop: - %d\n", data);  
75     pop (stack, &top, &data);  
76     printf ("pop: - %d\n", data);  
77     pop (stack, &top, &data);  
78     printf ("pop: - %d\n", data);  
79     return 0;  
80 }
```



## ex8-2.c

```

1  /* code: ex8-2.c    (v1.20.00) */
2  #include<stdio.h>
3  #include<stdlib.h>
4
5  #define MAX 128
6  #define ENQUEUE_SUCCESS      1
7  #define ENQUEUE_FAILURE     -1
8  #define DEQUEUE_SUCCESS      2
9  #define DEQUEUE_FAILURE     -2
10
11 /* ----- */
12 void queue_init (int *front, int *rear)
13 {
14     *front = -1;
15     *rear = -1;
16 }
17
18 /* ----- */
19 int enqueue (int q[], int *rear, int data)
20 {
21     if (*rear < MAX - 1) {
22         *rear = *rear + 1;
23         q[*rear] = data;
24         return ENQUEUE_SUCCESS;
25     }
26     else {
27         return ENQUEUE_FAILURE;
28     }
29 }
30
31 /* ----- */
32 int dequeue (int q[], int *front, int rear, int *data)
33 {
34     if (*front == rear) {
35         return DEQUEUE_FAILURE;

```

```
36     }
37     *front = *front + 1;
38     *data = q[*front];
39     return DEQUEUE_SUCCESS;
40 }
41
42 /* ----- */
43 int main ()
44 {
45     int queue[MAX];
46     int front, rear, data;
47     int stat;
48
49     queue_init (&front, &rear);
50     enqueue (queue, &rear, 100);
51     enqueue (queue, &rear, 200);
52     enqueue (queue, &rear, 300);
53     enqueue (queue, &rear, 400);
54     enqueue (queue, &rear, 500);
55     while (rear - front) {
56         stat = dequeue (queue, &front, rear, &data);
57         if (stat == DEQUEUE_SUCCESS) {
58             printf ("%d\n", data);
59         }
60         else {
61             printf ("QUEUE is empty\n");
62         }
63     }
64     return 0;
65 }
```

**ex8-3.c**

```

1  /* code: ex8-3.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <stdlib.h>
4  #define WIDTH  40
5  #define HEIGHT 20
6
7  /* ----- */
8  void cell_evolve (int array[HEIGHT][WIDTH])
9  {
10     int array_new[HEIGHT][WIDTH];
11     int x, y, n, x_width, y_height;
12
13     for (y = 0; y < HEIGHT; y++) {
14         for (x = 0; x < WIDTH; x++) {
15             n = 0;
16             for (y_height = y - 1; y_height <= y + 1;
17                  y_height++) {
18                 for (x_width = x - 1; x_width <= x + 1; x_width
19                      ++){
20                     if (array[(y_height + HEIGHT) % HEIGHT][(
21                          x_width + WIDTH) % WIDTH]) {
22                         n++;
23                     }
24                 }
25             }
26             if (array[y][x]) {
27                 n--;
28             }
29             array_new[y][x] = (n == 3 || (n == 2 && array[y][
30                  x]));
31         }
32     }
33     for (y = 0; y < HEIGHT; y++) {
34         for (x = 0; x < WIDTH; x++) {

```

```

32     array[y][x] = array_new[y][x];
33 }
34 }
35 }
36
37 /* ----- */
38 void cell_first_generation (int array[HEIGHT][WIDTH])
39 {
40     int x, y, r;
41     for (x = 0; x < WIDTH; x++) {
42         for (y = 0; y < HEIGHT; y++) {
43             r = RANDMAX / 8;
44             if (rand () < r) {
45                 array[y][x] = 1;
46             }
47             else {
48                 array[y][x] = 0;
49             }
50         }
51     }
52 }
53
54 /* ----- */
55 void cell_print (int array[HEIGHT][WIDTH], int
56                 generation)
57 {
58     int x, y;
59     printf (" [ Generation: -%05d]\n", generation);
60     for (y = 0; y < HEIGHT; y++) {
61         for (x = 0; x < WIDTH; x++) {
62             if (array[y][x] == 1) {
63                 printf ("*");
64             }
65             else {
66                 printf (".");
67             }
68         }
69     }
70     printf ("\n");

```

```
69     }
70     printf ("\n");
71     fflush (stdout);
72 }
73
74 /* ----- */
75 int main ()
76 {
77     int i;
78     int array[HEIGHT][WIDTH];
79     cell_first_generation (array);
80     i = 0;
81     while (i < 100) {
82         cell_print (array, i);
83         cell_evolve (array);
84         i++;
85     }
86     return 0;
87 }
```

## q8-1.c

```

1  /* code: q8-1.c    (v1.20.00) */
2  #include<stdio.h>
3  #include<stdlib.h>
4
5  #define MAX 128                      /* ring buffer max size
        */
6
7  #define ENQUEUE_SUCCESS      1
8  #define ENQUEUE_FAILURE     -1
9  #define DEQUEUE_SUCCESS      2
10 #define DEQUEUE_FAILURE     -2
11
12 /* ----- */
13 void rb_queue_init (int *front , int *rear)
14 {
15     *front = 0;
16     *rear = 0;
17 }
18
19 /* ----- */
20 int rb_enqueue (int q[] , int *front , int *rear , int
        data)
21 {
22     int index_f , index_r , index_q;
23     index_f = *front % MAX;
24     index_r = (*rear + 1) % MAX;
25     if (index_f != index_r) {
26         index_q = (*rear)++ % MAX;
27         q[index_q] = data;
28         return ENQUEUE_SUCCESS;
29     }
30     else {
31         return ENQUEUE_FAILURE;
32     }
33 }

```

```

34
35 /* ----- */
36 int rb_dequeue (int q[], int *front, int *rear, int *
    data)
37 {
38     int index;
39     if (*front != *rear) {
40         index = (*front)++ % MAX;
41         *data = q[index];
42         return DEQUEUE_SUCCESS;
43     }
44     else {
45         return DEQUEUE_FAILURE;
46     }
47 }
48
49 /* ----- */
50 int main ()
51 {
52     int queue[MAX];
53     int front, rear, data;
54     int stat;
55
56     rb_queue_init (&front, &rear);
57
58     rb_enqueue (queue, &front, &rear, 100);
59     rb_enqueue (queue, &front, &rear, 200);
60     rb_enqueue (queue, &front, &rear, 300);
61     rb_enqueue (queue, &front, &rear, 400);
62     rb_enqueue (queue, &front, &rear, 500);
63
64     while (rear - front) {
65         stat = rb_dequeue (queue, &front, &rear, &data);
66         if (stat == DEQUEUE_SUCCESS) {
67             printf ("%d\n", data);
68         }
69         else {
70             printf ("QUEUE is empty\n");

```

```
71     }  
72   }  
73  
74   return 0;  
75 }
```



## q8-2.c

```

1  /* code: q8-2.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <unistd.h>
5
6  #define WIDTH  40
7  #define HEIGHT 20
8
9  /* ----- */
10 void cell_evolve (int array[HEIGHT][WIDTH])
11 {
12     int array_new[HEIGHT][WIDTH];
13     int x, y, n, x_width, y_height;
14
15     for (y = 0; y < HEIGHT; y++) {
16         for (x = 0; x < WIDTH; x++) {
17             n = 0;
18             for (y_height = y - 1; y_height <= y + 1;
19                  y_height++) {
20                 for (x_width = x - 1; x_width <= x + 1; x_width
21                      ++){
22                     if (array[(y_height + HEIGHT) % HEIGHT][(
23                          x_width + WIDTH) % WIDTH]) {
24                         n++;
25                     }
26                 }
27             }
28             if (array[y][x]) {
29                 n--;
30             }
31             array_new[y][x] = (n == 3 || (n == 2 && array[y][
32                 x]));
33         }
34     }
35 }

```

```

32     for (y = 0; y < HEIGHT; y++) {
33         for (x = 0; x < WIDTH; x++) {
34             array[y][x] = array_new[y][x];
35         }
36     }
37 }
38
39 /* ----- */
40 void cell_first_generation (int array[HEIGHT][WIDTH])
41 {
42     int x, y, r;
43     for (x = 0; x < WIDTH; x++) {
44         for (y = 0; y < HEIGHT; y++) {
45             r = RANDMAX / 8;
46             if (rand () < r) {
47                 array[y][x] = 1;
48             }
49             else {
50                 array[y][x] = 0;
51             }
52         }
53     }
54 }
55
56
57 /* ----- */
58 void cell_print (int array[HEIGHT][WIDTH], int
59                 generation)
60 {
61     int x, y;
62     printf (" [ Generation: -%05d]\n", generation);
63     for (y = 0; y < HEIGHT; y++) {
64         for (x = 0; x < WIDTH; x++) {
65             if (array[y][x] == 1) {
66                 printf ("*");
67             }
68             else {

```

```

69         printf (".");
70     }
71 }
72     printf ("\n");
73 }
74     printf ("\n");
75     fflush (stdout);
76 }
77
78
79 /* ----- */
80 void cell_print_esc (int array[HEIGHT][WIDTH], int
    generation)
81 {
82     int x, y;
83
84     printf ("\e[H");
85     for (y = 0; y < HEIGHT; y++) {
86         for (x = 0; x < WIDTH; x++) {
87             if (array[y][x] == 1) {
88                 printf ("\e[07m- \e[m");
89             }
90             else {
91                 printf (" -");
92             }
93         }
94         printf ("\e[E");
95     }
96     fflush (stdout);
97 }
98
99 /* ----- */
100 int main ()
101 {
102     int i;
103     int array[HEIGHT][WIDTH];
104
105     cell_first_generation (array);

```

```
106     i = 0;
107     while (i < 100) {
108         cell_print_esc (array , i);
109         cell_evolve (array);
110         i++;
111         sleep (1);
112     }
113
114     return 0;
115 }
```

## ex9-1.c

```
1  /* code: ex9-1.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  int main ()
6  {
7      FILE *fptr;
8      fptr = fopen ("ex9-1-output.txt", "w");
9      fprintf (fptr, "The Open University of Japan\n");
10     fclose (fptr);
11     return 0;
12 }
```

## ex9-2.c

```
1  /* code: ex9-2.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  int main ()
6  {
7      FILE *fptr;
8      if (NULL == (fptr = fopen ("ex9-2-output.txt", "w")))
9      {
10         fprintf (stderr, "ERROR: Can not open file [output2
11             .txt]");
12         exit (-1);
13     }
14     fprintf (fptr, "The Open University of Japan\n");
15     fclose (fptr);
16     return 0;
17 }
```

## ex9-3.c

```

1  /* code: ex9-3.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  #define IRIS_DATA "iris.dat"
6
7  int main ()
8  {
9      FILE *fptr;
10     float sl, sw, pl, pw;
11     float s_sl, s_sw, s_pl, s_pw;
12     char name[128];
13     int n;
14
15     if (NULL == (fptr = fopen (IRIS_DATA, "r"))) {
16         fprintf (stderr, "ERROR: -Can not open file -[%s]",
17                 IRIS_DATA);
18         exit (-1);
19     }
20     n = 0;
21     s_sl = s_sw = s_pl = s_pw = 0.0;
22     while (EOF != fscanf (fptr, "%f,%f,%f,%f,%s", &sl, &
23         sw, &pl, &pw, name)) {
24         s_sl += sl;
25         s_sw += sw;
26         s_pl += pl;
27         s_pw += pw;
28         n++;
29     }
30     printf ("iris -data -: -%d\n", n);
31     printf ("avg. -sepal -length: -%f\n", s_sl / (float) n);
32     printf ("avg. -sepal -width -: -%f\n", s_sw / (float) n);
33     printf ("avg. -petal -length: -%f\n", s_pl / (float) n);
34     printf ("avg. -petal -width -: -%f\n", s_pw / (float) n);
35     fclose (fptr);

```

```
34 |  
35 |   return 0;  
36 | }
```



## q9-1.c

```

1  /* code: q9-1.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  #define IRIS_DATA "iris.dat"
6  #define MAXARRAY 256
7
8  struct iris
9  {
10     float sl, sw, pl, pw;
11     char name[128];
12 };
13 typedef struct iris IRIS_TYPE;
14
15 /* ----- */
16 int read_iris_data (IRIS_TYPE data[], int num, char *
    filename)
17 {
18     FILE *fptr;
19     float sl, sw, pl, pw;
20     char name[128];
21     int n;
22     if (NULL == (fptr = fopen (filename, "r"))) {
23         fprintf (stderr, "ERROR: Can not open file [%s]",
            filename);
24         exit (-1);
25     }
26     n = 0;
27     while (EOF != fscanf (fptr, "%f,%f,%f,%f,%s", &sl, &
        sw, &pl, &pw, name)) {
28         data[n].sl = sl;
29         data[n].sw = sw;
30         data[n].pl = pl;
31         data[n].pw = pw;
32         sprintf (data[n].name, "%s", name);

```

```

33     n++;
34     if (n >= num) {
35         fprintf (stderr, "ERROR: Not enough array size");
36         exit (-1);
37     }
38 }
39 fclose (fptr);
40 return n;
41 }
42
43 /* ----- */
44 IRIS_TYPE find_iris_avg (IRIS_TYPE data[], int num)
45 {
46     IRIS_TYPE avg;
47     int i;
48     float s_sl, s_sw, s_pl, s_pw;
49
50     s_sl = s_sw = s_pl = s_pw = 0.0;
51     for (i = 0; i < num; i++) {
52         s_sl += data[i].sl;
53         s_sw += data[i].sw;
54         s_pl += data[i].pl;
55         s_pw += data[i].pw;
56     }
57
58     avg.sl = s_sl / (float) num;
59     avg.sw = s_sw / (float) num;
60     avg.pl = s_pl / (float) num;
61     avg.pw = s_pw / (float) num;
62     sprintf (avg.name, "%s", "average");
63     return avg;
64 }
65
66 /* ----- */
67 int main ()
68 {
69     int num;
70     IRIS_TYPE iris_data[MAXARRAY];

```

```
71  IRIS_TYPE avg;  
72  
73  num = read_iris_data (iris_data , MAX_ARRAY, IRIS_DATA  
    );  
74  avg = find_iris_avg (iris_data , num);  
75  printf ("iris -data:-%d\n", num);  
76  printf ("avg.-sepal-length:-%f\n", avg.sl);  
77  printf ("avg.-sepal-width:-%f\n", avg.sw);  
78  printf ("avg.-petal-length:-%f\n", avg.pl);  
79  printf ("avg.-petal-width:-%f\n", avg.pw);  
80  
81  return 0;  
82 }
```

## q9-2.c

```

1  /* code: q9-2.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <string.h>
5
6  #define ROW  256
7  #define COL  256
8  #define FILTER_SIZE 3
9
10 struct pgm
11 {
12     int row;
13     int col;
14     int max;
15     float pixel[ROW][COL];
16 };
17 typedef struct pgm PGM_TYPE;
18
19
20 /* ----- */
21 void pgm_read (PGM_TYPE *image, char *filename)
22 {
23     FILE *infile;
24     int i, j;
25     char magic_number[32];
26
27     if (NULL == (infile = fopen (filename, "r"))) {
28         fprintf (stderr, "Can not open file [%s]", filename
29             );
30         exit (-1);
31     }
32     fscanf (infile, "%s", magic_number);
33     fscanf (infile, "%d", &image->col);
34     fscanf (infile, "%d", &image->row);
35     fscanf (infile, "%d", &image->max);

```

```

35     printf ("image:%s-[%dx%d]-(%d)\n",
36             magic_number, image->col, image->row, image->
                max);
37     if (strcmp ("P2", magic_number)) {
38         fprintf (stderr, "Not PGM(P2) - file!");
39         exit (-2);
40     }
41     for (i = 0; i < image->row; i++) {
42         for (j = 0; j < image->col; j++) {
43             fscanf (infile, "%f", &image->pixel[i][j]);
44         }
45     }
46     fclose (infile);
47 }
48
49 /* _____ */
50 void pgm_write (PGMLTYPE *image, char *filename)
51 {
52     FILE *outfile;
53     int i, j;
54
55     if (NULL == (outfile = fopen (filename, "w"))) {
56         fprintf (stderr, "Can not open file [%s]", filename
                    );
57         exit (-1);
58     }
59     fprintf (outfile, "%s\n", "P2");
60     fprintf (outfile, "%d-", image->col);
61     fprintf (outfile, "%d-\n", image->row);
62     fprintf (outfile, "%d-\n", image->max);
63
64     for (i = 0; i < image->row; i++) {
65         for (j = 0; j < image->col; j++) {
66             fprintf (outfile, "%2d-", (int) image->pixel[i][j
                ]);
67         }
68         fprintf (outfile, "\n");
69     }

```

```

70     fclose (outfile);
71 }
72
73 /* ----- */
74 void convolution (PGM_TYPE *image_input, PGM_TYPE *
    image_output)
75 {
76     int i, j, k, l, px, py, sum;
77     float dsum;
78
79     float filter[FILTER_SIZE][FILTER_SIZE] = {
80         {-1.0, -1.0, -1.0},
81         {-1.0, +8.0, -1.0},
82         {-1.0, -1.0, -1.0}
83     };
84
85     image_output->row = image_input->row;
86     image_output->col = image_input->col;
87     image_output->max = image_input->max;
88     px = (FILTER_SIZE - 1) / 2;
89     py = (FILTER_SIZE - 1) / 2;
90     dsum = 0;
91     for (i = 0 + px; i < image_input->row - px; i++) {
92         for (j = 0 + py; j < image_input->col - py; j++) {
93             dsum = 0.0;
94             for (k = 0; k < FILTER_SIZE; k++) {
95                 for (l = 0; l < FILTER_SIZE; l++) {
96                     dsum += filter[k][l] * image_input->pixel[i -
                        px + k][j - py + l];
97                 }
98             }
99             sum = (int) dsum;
100             if (sum > image_input->max) {
101                 sum = image_input->max;
102             }
103             if (sum < 0) {
104                 sum = 0;
105             }

```

```
106     image_output->pixel[i][j] = sum;
107 }
108 }
109 }
110
111 /* _____ */
112 int main ()
113 {
114     PGM_TYPE *pgm_input;
115     PGM_TYPE *pgm_output;
116
117     pgm_input = malloc (sizeof (PGM_TYPE));
118     pgm_output = malloc (sizeof (PGM_TYPE));
119
120     pgm_read (pgm_input, "sample.pgm");
121     convolution (pgm_input, pgm_output);
122     pgm_write (pgm_output, "q9-2-laplacian8.pgm");
123
124     free (pgm_input);
125     free (pgm_output);
126
127     return 0;
128 }
```

## q9-3.c

```

1  /* code: q9-3.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <string.h>
5
6  #define ROW  256
7  #define COL  256
8  #define FILTER_SIZE 3
9
10 struct pgm
11 {
12     int row;
13     int col;
14     int max;
15     float pixel[ROW][COL];
16 };
17 typedef struct pgm PGM_TYPE;
18
19
20 /* ----- */
21 void pgm_read (PGM_TYPE *image, char *filename)
22 {
23     FILE *infile;
24     int i, j;
25     char magic_number[32];
26
27     if (NULL == (infile = fopen (filename, "r"))) {
28         fprintf (stderr, "Can not open file [%s]", filename
29             );
30         exit (-1);
31     }
32     fscanf (infile, "%s", magic_number);
33     fscanf (infile, "%d", &image->col);
34     fscanf (infile, "%d", &image->row);
35     fscanf (infile, "%d", &image->max);

```



```

35     printf ("image:%s-[%dx%d]-(%d)\n",
36             magic_number, image->col, image->row, image->
                max);
37     if (strcmp ("P2", magic_number)) {
38         fprintf (stderr, "Not PGM(P2) - file!");
39         exit (-2);
40     }
41     for (i = 0; i < image->row; i++) {
42         for (j = 0; j < image->col; j++) {
43             fscanf (infile, "%f", &image->pixel[i][j]);
44         }
45     }
46     fclose (infile);
47 }
48
49 /* _____ */
50 void pgm_write (PGMLTYPE *image, char *filename)
51 {
52     FILE *outfile;
53     int i, j;
54
55     if (NULL == (outfile = fopen (filename, "w"))) {
56         fprintf (stderr, "Can not open file [%s]", filename
                    );
57         exit (-1);
58     }
59     fprintf (outfile, "%s\n", "P2");
60     fprintf (outfile, "%d-", image->col);
61     fprintf (outfile, "%d-\n", image->row);
62     fprintf (outfile, "%d-\n", image->max);
63
64     for (i = 0; i < image->row; i++) {
65         for (j = 0; j < image->col; j++) {
66             fprintf (outfile, "%2d-", (int) image->pixel[i][j
                ]);
67         }
68         fprintf (outfile, "\n");
69     }

```

```

70     fclose (outfile);
71 }
72
73 /* ----- */
74 void convolution (PGM_TYPE *image_input, PGM_TYPE *
75     image_output)
76 {
77     int i, j, k, l, px, py, sum;
78     float dsum;
79
80     float filter[FILTER_SIZE][FILTER_SIZE] = {
81         {0.0, -1.0, -2.0},
82         {1.0, 0.0, -1.0},
83         {2.0, 1.0, 0.0}
84     };
85
86     image_output->row = image_input->row;
87     image_output->col = image_input->col;
88     image_output->max = image_input->max;
89     px = (FILTER_SIZE - 1) / 2;
90     py = (FILTER_SIZE - 1) / 2;
91     dsum = 0;
92     for (i = 0 + px; i < image_input->row - px; i++) {
93         for (j = 0 + py; j < image_input->col - py; j++) {
94             dsum = 0.0;
95             for (k = 0; k < FILTER_SIZE; k++) {
96                 for (l = 0; l < FILTER_SIZE; l++) {
97                     dsum += filter[k][l] * image_input->pixel[i -
98                         px + k][j - py + l];
99                 }
100             }
101             sum = (int) dsum;
102             if (sum > image_input->max) {
103                 sum = image_input->max;
104             }
105             if (sum < 0) {
106                 sum = 0;
107             }

```

```
106     image_output->pixel[i][j] = sum;
107 }
108 }
109 }
110
111 /* _____ */
112 int main ()
113 {
114     PGM_TYPE *pgm_input;
115     PGM_TYPE *pgm_output;
116
117     pgm_input = malloc (sizeof (PGM_TYPE));
118     pgm_output = malloc (sizeof (PGM_TYPE));
119
120     pgm_read (pgm_input, "sample.pgm");
121     convolution (pgm_input, pgm_output);
122     pgm_write (pgm_output, "q9-3-sobel-d.pgm");
123
124     free (pgm_input);
125     free (pgm_output);
126
127     return 0;
128 }
```

## q9-4.c

```

1  /* code: q9-4.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <string.h>
5
6  #define ROW  256
7  #define COL  256
8  #define FILTER_SIZE 3
9
10 struct pgm
11 {
12     int row;
13     int col;
14     int max;
15     float pixel[ROW][COL];
16 };
17 typedef struct pgm PGM_TYPE;
18
19
20 /* ----- */
21 void pgm_read (PGM_TYPE *image, char *filename)
22 {
23     FILE *infile;
24     int i, j;
25     char magic_number[32];
26
27     if (NULL == (infile = fopen (filename, "r"))) {
28         fprintf (stderr, "Can not open file [%s]", filename
29             );
30         exit (-1);
31     }
32     fscanf (infile, "%s", magic_number);
33     fscanf (infile, "%d", &image->col);
34     fscanf (infile, "%d", &image->row);
35     fscanf (infile, "%d", &image->max);

```

```

35     printf ("image:%s-[%dx%d]-(%d)\n",
36             magic_number, image->col, image->row, image->
                max);
37     if (strcmp ("P2", magic_number)) {
38         fprintf (stderr, "Not PGM(P2) file!");
39         exit (-2);
40     }
41     for (i = 0; i < image->row; i++) {
42         for (j = 0; j < image->col; j++) {
43             fscanf (infile, "%f", &image->pixel[i][j]);
44         }
45     }
46     fclose (infile);
47 }
48
49 /* _____ */
50 void pgm_write (PGMLTYPE *image, char *filename)
51 {
52     FILE *outfile;
53     int i, j;
54
55     if (NULL == (outfile = fopen (filename, "w"))) {
56         fprintf (stderr, "Can not open file [%s]", filename
                    );
57         exit (-1);
58     }
59     fprintf (outfile, "%s\n", "P2");
60     fprintf (outfile, "%d-", image->col);
61     fprintf (outfile, "%d-\n", image->row);
62     fprintf (outfile, "%d-\n", image->max);
63
64     for (i = 0; i < image->row; i++) {
65         for (j = 0; j < image->col; j++) {
66             fprintf (outfile, "%2d-", (int) image->pixel[i][j
                ]);
67         }
68         fprintf (outfile, "\n");
69     }

```

```

70     fclose (outfile);
71 }
72
73 /* ----- */
74 void convolution (PGM_TYPE *image_input, PGM_TYPE *
75     image_output)
76 {
77     int i, j, k, l, px, py, sum;
78     float dsum;
79
80     float filter[FILTER_SIZE][FILTER_SIZE] =
81         { {1.00 / 16.00, 2.00 / 16.00, 1.00 / 16.00},
82           {2.00 / 16.00, 4.00 / 16.00, 2.00 / 16.00},
83           {1.00 / 16.00, 2.00 / 16.00, 1.00 / 16.00}
84         };
85
86     image_output->row = image_input->row;
87     image_output->col = image_input->col;
88     image_output->max = image_input->max;
89     px = (FILTER_SIZE - 1) / 2;
90     py = (FILTER_SIZE - 1) / 2;
91     dsum = 0;
92     for (i = 0 + px; i < image_input->row - px; i++) {
93         for (j = 0 + py; j < image_input->col - py; j++) {
94             dsum = 0.0;
95             for (k = 0; k < FILTER_SIZE; k++) {
96                 for (l = 0; l < FILTER_SIZE; l++) {
97                     dsum += filter[k][l] * image_input->pixel[i -
98                         px + k][j - py + l];
99                 }
100             }
101             sum = (int) dsum;
102             if (sum > image_input->max) {
103                 sum = image_input->max;
104             }
105             if (sum < 0) {
106                 sum = 0;
107             }

```

```
106     image_output->pixel[i][j] = sum;
107 }
108 }
109 }
110
111 /* ----- */
112 int main ()
113 {
114     PGM_TYPE *pgm_input;
115     PGM_TYPE *pgm_output;
116
117     pgm_input = malloc (sizeof (PGM_TYPE));
118     pgm_output = malloc (sizeof (PGM_TYPE));
119
120     pgm_read (pgm_input, "sample.pgm");
121     convolution (pgm_input, pgm_output);
122     pgm_write (pgm_output, "q9-4-gaussian.pgm");
123
124     free (pgm_input);
125     free (pgm_output);
126
127     return 0;
128 }
```

## ex10-1.c

```

1  /* code: ex10-1.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <stdlib.h>
4  /* ----- */
5  void print_array (int v[], int n)
6  {
7      int i;
8      printf ("array:-");
9      for (i = 0; i < n; i++) {
10         printf ("%d-", v[i]);
11     }
12     printf ("\n");
13 }
14
15 /* ----- */
16 void bubble_sort (int v[], int n)
17 {
18     int i, j, t;
19     for (i = 0; i < n - 1; i++) {
20         for (j = n - 1; j > i; j--) {
21             if (v[j - 1] > v[j]) {
22                 t = v[j];
23                 v[j] = v[j - 1];
24                 v[j - 1] = t;
25             }
26             printf ("i:%d-j:%d-", i, j);
27             print_array (v, n);
28         }
29     }
30 }
31
32 /* ----- */
33 int main ()
34 {
35     int array[5]

```



```
36  = { 30, 50, 20, 10, 40 };  
37  print_array (array , 5);  
38  bubble_sort (array , 5);  
39  print_array (array , 5);  
40  return 0;  
41  }
```

## ex10-2.c

```

1  /* code: ex10-2.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  /* ----- */
6  void print_array (int v[], int n)
7  {
8      int i;
9      printf ("array:-");
10     for (i = 0; i < n; i++) {
11         printf ("%d-", v[i]);
12     }
13     printf ("\n");
14 }
15
16 /* ----- */
17 void selection_sort (int v[], int n)
18 {
19     int i, j, t, min_index;
20     for (i = 0; i < n - 1; i++) {
21         min_index = i;
22         for (j = i + 1; j < n; j++) {
23             if (v[j] < v[min_index]) {
24                 min_index = j;
25             }
26             printf ("i:%d-j:%d-", i, j);
27             print_array (v, n);
28         }
29         t = v[i];
30         v[i] = v[min_index];
31         v[min_index] = t;
32     }
33 }
34
35 /* ----- */

```

```
36 int main ()
37 {
38     int array[5]
39     = { 30, 50, 20, 10, 40 };
40     print_array (array, 5);
41     selection_sort (array, 5);
42     print_array (array, 5);
43     return 0;
44 }
```

## ex10-3.c

```

1  /* code: ex10-3.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  /* ----- */
6  void print_array (int v[], int n)
7  {
8      int i;
9      printf ("array:-");
10     for (i = 0; i < n; i++) {
11         printf ("%d-", v[i]);
12     }
13     printf ("\n");
14 }
15
16 /* ----- */
17 void insertion_sort (int v[], int n)
18 {
19     int i, j, t;
20     for (i = 1; i < n; i++) {
21         j = i;
22         while ((j >= 1) && (v[j - 1] > v[j])) {
23             t = v[j];
24             v[j] = v[j - 1];
25             v[j - 1] = t;
26             j--;
27             printf ("i:%d-j:%d-", i, j);
28             print_array (v, n);
29         }
30     }
31 }
32
33 /* ----- */
34 int main ()
35 {

```

```
36  int array[5]
37  = { 30, 50, 20, 10, 40 };
38  print_array (array, 5);
39  insertion_sort (array, 5);
40  print_array (array, 5);
41  return 0;
42 }
```

## ex10-4.c

```

1  /* code: ex10-4.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  /* ----- */
6  void print_array (int v[], int n)
7  {
8      int i;
9      printf ("array:-");
10     for (i = 0; i < n; i++) {
11         printf ("%d-", v[i]);
12     }
13     printf ("\n");
14 }
15
16 /* ----- */
17 void insertion_sort (int v[], int n)
18 {
19     int i, j, t;
20     for (i = 1; i < n; i++) {
21         j = i;
22         while ((j >= 1) && (v[j - 1] > v[j])) {
23             t = v[j];
24             v[j] = v[j - 1];
25             v[j - 1] = t;
26             j--;
27             printf ("i:%d-j:%d-", i, j);
28             print_array (v, n);
29         }
30     }
31 }
32
33 /* ----- */
34 int main ()
35 {

```

```
36  int array[5]
37  = { 10, 20, 30, 40, 50 };
38  print_array (array, 5);
39  insertion_sort (array, 5);
40  print_array (array, 5);
41  return 0;
42 }
```

## ex10-5.c

```

1  /* code: ex10-5.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  /* ----- */
6  void print_array (int v[], int n)
7  {
8      int i;
9      printf ("array:-");
10     for (i = 0; i < n; i++) {
11         printf ("%d-", v[i]);
12     }
13     printf ("\n");
14 }
15
16 /* ----- */
17 void insertion_sort (int v[], int n)
18 {
19     int i, j, t;
20     for (i = 1; i < n; i++) {
21         j = i;
22         while ((j >= 1) && (v[j - 1] > v[j])) {
23             t = v[j];
24             v[j] = v[j - 1];
25             v[j - 1] = t;
26             j--;
27             printf ("i:%d-j:%d-", i, j);
28             print_array (v, n);
29         }
30     }
31 }
32
33 /* ----- */
34 int main ()
35 {

```



```
36 |   int array[5] = { 50, 40, 30, 20, 10 };  
37 |   print_array (array , 5);  
38 |   insertion_sort (array, 5);  
39 |   print_array (array , 5);  
40 |   return 0;  
41 | }
```

## q10-1.c

```

1  /* code: q10-1.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  /* ----- */
6  void print_array (int v[], int n)
7  {
8      int i;
9      printf ("array:-");
10     for (i = 0; i < n; i++) {
11         printf ("%d-", v[i]);
12     }
13     printf ("\n");
14 }
15
16 /* ----- */
17 void bubble_sort (int v[], int n)
18 {
19     int i, j, t, flag;
20
21     flag = 1;
22     for (i = 0; (i < n - 1) && (flag == 1); i++) {
23         flag = 0;
24         for (j = n - 1; j > i; j--) {
25             if (v[j - 1] > v[j]) {
26                 t = v[j];
27                 v[j] = v[j - 1];
28                 v[j - 1] = t;
29                 flag = 1;
30             }
31             printf ("i:%d-j:%d-", i, j);
32             print_array (v, n);
33         }
34     }
35 }

```

```
36
37 /* ----- */
38 int main ()
39 {
40     /*  int array[5]
41         = { 30, 50, 20, 10, 40 }; */
42     /*  int array[5]
43         = { 50, 40, 30, 20, 10 }; */
44
45     int array[5]
46     = { 10, 20, 30, 40, 50 };
47
48     print_array (array , 5);
49     bubble_sort (array , 5);
50     print_array (array , 5);
51
52     return 0;
53 }
```

## q10-2.c

```
1  /* code: q10-2.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  /* ----- */
6  void print_array (int v[], int n)
7  {
8      int i;
9      printf ("array:-");
10     for (i = 0; i < n; i++) {
11         printf ("%d-", v[i]);
12     }
13     printf ("\n");
14 }
15
16 /* ----- */
17 int int_compare (const void *va, const void *vb)
18 {
19     int a, b;
20     a = *(int *) va;
21     b = *(int *) vb;
22     if (a < b) {
23         return (-1);
24     }
25     else if (a > b) {
26         return (1);
27     }
28     else {
29         return (0);
30     }
31 }
32
33 /* ----- */
34 int main ()
35 {
```

```
36  int array[5]
37  = { 30, 50, 20, 10, 40 };
38
39  print_array (array , 5);
40  qsort (array , 5, sizeof (int), int_compare);
41  print_array (array , 5);
42
43  return 0;
44 }
```

## q10-3.c

```

1  /* code: q10-3.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <string.h>
5
6  /* ----- */
7  int cmp_string (const void *p1, const void *p2)
8  {
9      return strcmp (*(char *const *) p1, *(char *const *)
10                     p2);
11 }
12
13 /* ----- */
14 void print_str_array (char *v[], int n)
15 {
16     int i;
17     printf ("array:-");
18     for (i = 0; i < n; i++) {
19         printf ("%s-", v[i]);
20     }
21     printf ("\n");
22 }
23
24 /* ----- */
25 int main ()
26 {
27     char *array[7] = {
28         "Sunday", "Monday", "Tuesday",
29         "Wednesday", "Thursday", "Friday", "Saturday",
30     };
31     print_str_array (array, 7);
32     qsort (array, 7, sizeof (char *), cmp_string);
33     print_str_array (array, 7);
34     return 0;
35 }

```

## ex11-1.c

```
1  /* code: ex11-1.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  /* ----- */
6  void print_array (int v[], int n)
7  {
8      int i;
9      printf ("array:-");
10     for (i = 0; i < n; i++) {
11         printf ("%d-", v[i]);
12     }
13     printf ("\n");
14 }
15
16 /* ----- */
17 int partition (int v[], int lower_bound, int
    upper_bound)
18 {
19     int a, down, up, temp;
20
21     a = v[lower_bound];
22     up = upper_bound;
23     down = lower_bound;
24
25     while (down < up) {
26         while ((v[down] <= a) && (down < upper_bound)) {
27             down++;
28         }
29         while (v[up] > a) {
30             up--;
31         }
32         if (down < up) {
33             temp = v[down];
34             v[down] = v[up];
```

```
35     v[up] = temp;
36 }
37 }
38 v[lower_bound] = v[up];
39 v[up] = a;
40 return up;
41 }
42
43 /* ----- */
44 void quicksort (int v[], int left, int right)
45 {
46     int p;
47     if (left >= right) {
48         return;
49     }
50     p = partition (v, left, right);
51     quicksort (v, left, p - 1);
52     quicksort (v, p + 1, right);
53 }
54
55 /* ----- */
56 int main ()
57 {
58     int array[10]
59     = { 80, 40, 30, 20, 10, 00, 70, 90, 50, 60 };
60
61     print_array (array, 10);
62     quicksort (array, 0, 9);
63     print_array (array, 10);
64
65     return 0;
66 }
```



## ex11-2.c

```
1  /* code: ex11-2.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <stdlib.h>
4  #define MAX 10
5
6  /* ----- */
7  void print_array (int v[], int n)
8  {
9      int i;
10     printf ("array:~");
11     for (i = 0; i < n; i++) {
12         printf ("%4d~", v[i]);
13     }
14     printf ("\n");
15 }
16
17 /* ----- */
18 void radix_sort (int a[], int n)
19 {
20     int i, max, exp;
21     int temp[MAX];
22     int bucket[10];
23
24     max = 0;
25     exp = 1;
26     for (i = 0; i < n; i++) {
27         if (a[i] > max) {
28             max = a[i];
29         }
30     }
31     while (max / exp > 0) {
32         for (i = 0; i < 10; i++) {
33             bucket[i] = 0;
34         }
35         for (i = 0; i < n; i++) {
```

```
36     bucket[a[i] / exp % 10]++;
37 }
38 for (i = 1; i < 10; i++) {
39     bucket[i] += bucket[i - 1];
40 }
41 for (i = n - 1; i >= 0; i--) {
42     temp[--bucket[a[i] / exp % 10]] = a[i];
43 }
44 for (i = 0; i < n; i++) {
45     a[i] = temp[i];
46 }
47 exp *= 10;
48 print_array (a, n);
49 }
50 }
51
52 /* ----- */
53 int main ()
54 {
55     int array[MAX]
56     = { 12, 19, 10, 28, 30, 01, 502, 16, 34, 177 };
57
58     print_array (array, 10);
59     radix_sort (array, 10);
60     print_array (array, 10);
61
62     return 0;
63 }
```

## q11-1.c

```
1  /* code: q11-1.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  /* ----- */
6  void print_array (int v[], int n)
7  {
8      int i;
9      printf ("array:-");
10     for (i = 0; i < n; i++) {
11         printf ("%d-", v[i]);
12     }
13     printf ("\n");
14 }
15
16 /* ----- */
17 int partition (int v[], int lower_bound, int
    upper_bound)
18 {
19     int a, down, up, temp;
20
21     a = v[lower_bound];
22     up = upper_bound;
23     down = lower_bound;
24
25     while (down < up) {
26         while ((v[down] <= a) && (down < upper_bound)) {
27             down++;
28         }
29         while (v[up] > a) {
30             up--;
31         }
32         if (down < up) {
33             temp = v[down];
34             v[down] = v[up];
```

```

35     v[up] = temp;
36 }
37 }
38 v[lower_bound] = v[up];
39 v[up] = a;
40 return up;
41 }
42
43 /* ----- */
44 void quicksort (int v[], int left, int right, int level
45 )
46 {
47     int p;
48     int i;
49     if (left >= right) {
50         return;
51     }
52     level++;
53     p = partition (v, left, right);
54     quicksort (v, left, p - 1, level);
55     quicksort (v, p + 1, right, level);
56     for (i = 0; i < level; i++) {
57         printf ("*");
58     }
59     printf (" [pivot:%d-left:%d-right:%d]\n", p, left,
60         right);
61 }
62 /* ----- */
63 int main ()
64 {
65     int array[10]
66     = { 30, 40, 80, 20, 10, 00, 70, 90, 50, 60 };
67
68     print_array (array, 10);
69     quicksort (array, 0, 9, 0);
70     print_array (array, 10);

```

```
71     return 0;  
72 }
```

## q11-2.c

```

1  /* code: q11-2.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <stdlib.h>
4  #define STACK_SIZE 2048
5
6  /* ----- */
7  void print_array (int v[], int n)
8  {
9      int i;
10     printf ("\n");
11     for (i = 0; i < n; i++) {
12         printf ("%02d-", v[i]);
13     }
14 }
15
16 /* ----- */
17 int partition (int v[], int lower_bound, int
    upper_bound)
18 {
19     int a, down, up, temp;
20
21     a = v[lower_bound];
22     up = upper_bound;
23     down = lower_bound;
24
25     while (down < up) {
26         while ((v[down] <= a) && (down < upper_bound)) {
27             down++;
28         }
29         while (v[up] > a) {
30             up--;
31         }
32         if (down < up) {
33             temp = v[down];
34             v[down] = v[up];

```

```

35     v[up] = temp;
36 }
37 }
38 v[lower_bound] = v[up];
39 v[up] = a;
40 return up;
41 }
42
43 /* ----- */
44 void quicksort_stack (int v[], int n)
45 {
46     int left, right, i, sptr;
47     int stack_lower_bound[STACK_SIZE];
48     int stack_upper_bound[STACK_SIZE];
49
50     stack_lower_bound[0] = 0;
51     stack_upper_bound[0] = n - 1;
52     sptr = 1;
53
54     while (sptr > 0) {
55         sptr--;
56         left = stack_lower_bound[sptr];
57         right = stack_upper_bound[sptr];
58
59         if (left >= right) {
60             ;
61         }
62         else {
63             i = partition (v, left, right);
64
65             if ((i - left) < (right - i)) {
66                 stack_lower_bound[sptr] = i + 1;
67                 stack_upper_bound[sptr++] = right;
68                 stack_lower_bound[sptr] = left;
69                 stack_upper_bound[sptr++] = i - 1;
70             }
71             else {
72                 stack_lower_bound[sptr] = left;

```

```
73         stack_upper_bound [sptr++] = i - 1;
74         stack_lower_bound [sptr] = i + 1;
75         stack_upper_bound [sptr++] = right;
76     }
77 }
78 }
79 }
80
81
82 /* _____ */
83 int main ()
84 {
85     int array[10]
86     = { 80, 40, 30, 20, 10, 00, 70, 90, 50, 60 };
87
88     print_array (array , 10);
89     quicksort_stack (array , 10);
90     print_array (array , 10);
91
92     return 0;
93 }
```



## q11-3.c

```

1  /* code: q11-3.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  #define MAX 100000000
6
7  /* ----- */
8  void print_array (int v[], int n)
9  {
10     int i;
11     printf ("array:-");
12     for (i = 0; i < n; i++) {
13         printf ("%8d-", v[i]);
14     }
15     printf ("\n");
16 }
17
18 /* ----- */
19 void rand_data (int v[], int n)
20 {
21     int i;
22     for (i = 0; i < n; i++) {
23         v[i] = rand () % (MAX / 10);
24     }
25 }
26
27 /* ----- */
28 int partition (int v[], int lower_bound, int
    upper_bound)
29 {
30     int a, down, up, temp;
31
32     a = v[lower_bound];
33     up = upper_bound;
34     down = lower_bound;

```

```

35
36   while (down < up) {
37       while ((v[down] <= a) && (down < upper_bound)) {
38           down++;
39       }
40       while (v[up] > a) {
41           up--;
42       }
43       if (down < up) {
44           temp = v[down];
45           v[down] = v[up];
46           v[up] = temp;
47       }
48   }
49   v[lower_bound] = v[up];
50   v[up] = a;
51   return up;
52 }
53
54 /* ----- */
55 void quicksort (int v[], int left, int right)
56 {
57     int p;
58     if (left >= right) {
59         return;
60     }
61     p = partition (v, left, right);
62     quicksort (v, left, p - 1);
63     quicksort (v, p + 1, right);
64 }
65
66 /* ----- */
67 int main (int argc, char **argv)
68 {
69     int *array;
70
71     array = malloc (sizeof (int) * MAX);
72     if (array == NULL) {

```

```
73     printf ("ERROR: Can not allocate memory\n");
74     exit (-1);
75 }
76
77 rand_data (array, MAX);
78 printf ("array size: %d\n", MAX);
79 fflush (stdout);
80
81 print_array (array, 30);
82
83 quicksort (array, 0, MAX - 1);
84
85 print_array (array, 30);
86
87 free (array);
88 return 0;
89 }
```

## q11-4.c

```

1  /* code: q11-4.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  /* ----- */
6  void print_array (int v[], int n)
7  {
8      int i;
9      printf ("\n");
10     for (i = 0; i < n; i++) {
11         printf ("%d-", v[i]);
12     }
13 }
14
15 /* ----- */
16 void merge_sort (int v[], int lb, int ub, int v_temp[])
17 {
18     int i, j, k, c;
19
20     if (lb >= ub) {
21         return;
22     }
23     c = (lb + ub) / 2;
24
25     merge_sort (v, lb, c, v_temp);
26     merge_sort (v, c + 1, ub, v_temp);
27
28     for (i = lb; i <= c; i++) {
29         v_temp[i] = v[i];
30     }
31     for (i = c + 1, j = ub; i <= ub; i++, j--) {
32         v_temp[i] = v[j];
33     }
34
35     i = lb;

```

```
36     j = ub;
37
38     for (k = lb; k <= ub; k++) {
39         if (v_temp[i] <= v_temp[j]) {
40             v[k] = v_temp[i++];
41         }
42         else {
43             v[k] = v_temp[j--];
44         }
45     }
46 }
47
48 /* ----- */
49 int main ()
50 {
51     int array[10] = { 8, 4, 3, 2, 1, 0, 7, 9, 5, 6 };
52     int array_temp[10];
53
54     print_array (array, 10);
55     merge_sort (array, 0, 9, array_temp);
56     print_array (array, 10);
57
58     return 0;
59 }
```

## ex12-1.c

```
1  /* code: ex12-1.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <stdlib.h>
4  #define ARRAY_SIZE 3000000
5
6  int main ()
7  {
8      int array[ARRAY_SIZE];
9      int i;
10     for (i = 0; i < ARRAY_SIZE; i++) {
11         array[i] = 100;
12     }
13     for (i = 0; i < 10; i++) {
14         printf ("%d-", array[i]);
15     }
16     return 0;
17 }
```

## ex12-2.c

```
1  /* code: ex12-2.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <stdlib.h>
4  #define ARRAY_SIZE 3000000
5
6  int array[ARRAY_SIZE];
7
8  int main ()
9  {
10     int i;
11     for (i = 0; i < ARRAY_SIZE; i++) {
12         array[i] = 100;
13     }
14     for (i = 0; i < 10; i++) {
15         printf ("%d-", array[i]);
16     }
17     return 0;
18 }
```

## ex12-3.c

```
1  /* code: ex12-3.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <stdlib.h>
4  #define ARRAY_SIZE 3000000
5
6  int main ()
7  {
8      int *array;
9      int i;
10
11     array = malloc (sizeof (int) * ARRAY_SIZE);
12
13     if (NULL == array) {
14         fprintf (stderr, "Error: - malloc () -\n");
15         exit (-1);
16     }
17     else {
18         for (i = 0; i < ARRAY_SIZE; i++) {
19             array[i] = 100;
20         }
21         for (i = 0; i < 10; i++) {
22             printf ("%d ", array[i]);
23         }
24         free (array);
25     }
26
27     return 0;
28 }
```



## ex12-4.c

```
1  /* code: ex12-4.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  /* _____ */
6  int main ()
7  {
8      int **array;
9      int i, j, rows, columns;
10
11     rows = 768;
12     columns = 1024;
13
14     array = malloc (rows * sizeof (int *));
15     for (i = 0; i < rows; i++) {
16         array[i] = malloc (columns * sizeof (int));
17     }
18
19     for (i = 0; i < rows; i++) {
20         for (j = 0; j < columns; j++) {
21             array[i][j] = rand () % 10;
22         }
23     }
24
25     for (i = 0; i < rows; i++) {
26         for (j = 0; j < columns; j++) {
27             printf ("%d-", array[i][j]);
28         }
29         printf ("\n");
30     }
31
32     for (i = 0; i < rows; i++) {
33         free (array[i]);
34     }
35     free (array);
```

```
36 |  
37 |   return 0;  
38 | }
```

## ex12-5.c

```
1  /* code: ex12-5.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  /* _____ */
6  int main ()
7  {
8      int **array;
9      int i, j, rows, columns;
10
11     rows = 768;
12     columns = 1024;
13
14     array = malloc (rows * sizeof (int *));
15     array[0] = malloc (rows * columns * sizeof (int));
16     for (i = 1; i < rows; i++) {
17         array[i] = array[0] + i * columns;
18     }
19
20     for (i = 0; i < rows; i++) {
21         for (j = 0; j < columns; j++) {
22             array[i][j] = rand () % 10;
23         }
24     }
25
26     for (i = 0; i < rows; i++) {
27         for (j = 0; j < columns; j++) {
28             printf ("%d-", array[i][j]);
29         }
30         printf ("\n");
31     }
32
33     free (array[0]);
34     free (array);
35
```

```
36     return 0;  
37 }
```

# q12-1.c

```

1  /* code: q12-1.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  /* _____ */
6  int main ()
7  {
8      int ***array;
9      int i, j, k;
10     int x, y, z;
11
12     x = 10;
13     y = 20;
14     z = 30;
15
16     array = malloc (x * sizeof (int **));
17     for (i = 0; i < x; i++) {
18         array[i] = malloc (y * sizeof (int *));
19         for (j = 0; j < y; j++) {
20             array[i][j] = malloc (z * sizeof (int));
21         }
22     }
23
24     for (i = 0; i < x; i++) {
25         for (j = 0; j < y; j++) {
26             for (k = 0; k < z; k++) {
27                 array[i][j][k] = rand () % 10;
28             }
29         }
30     }
31
32     for (i = 0; i < x; i++) {
33         for (j = 0; j < y; j++) {
34             for (k = 0; k < z; k++) {
35                 printf ("%d-", array[i][j][k]);

```

```
36     }
37   }
38 }
39
40 for (i = 0; i < x; i++) {
41   for (j = 0; j < y; j++) {
42     free (array[i][j]);
43   }
44   free (array[i]);
45 }
46 free (array);
47
48 return 0;
49 }
```

## q12-2.c

```
1  /* code: q12-2.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  /* ----- */
6  int **m2d_allocate (int rows, int columns)
7  {
8      int **array;
9      int i;
10     array = malloc (rows * sizeof (int *));
11     for (i = 0; i < rows; i++) {
12         array[i] = malloc (columns * sizeof (int));
13     }
14     return array;
15 }
16
17 /* ----- */
18 void m2d_deallocate (int **array, int rows)
19 {
20     int i;
21     for (i = 0; i < rows; i++) {
22         free (array[i]);
23     }
24     free (array);
25 }
26
27 /* ----- */
28 int main ()
29 {
30     int **array;
31     int i, j, rows, columns;
32
33     rows = 768;
34     columns = 1024;
35
```

```
36     array = m2d_allocate (rows, columns);
37
38     for (i = 0; i < rows; i++) {
39         for (j = 0; j < columns; j++) {
40             array[i][j] = rand () % 10;
41         }
42     }
43
44     for (i = 0; i < rows; i++) {
45         for (j = 0; j < columns; j++) {
46             printf ("%d-", array[i][j]);
47         }
48         printf ("\n");
49     }
50
51     m2d_deallocate (array, rows);
52
53     return 0;
54 }
```



## ex13-1.c

```

1  /* code: ex13-1.c    (v1.20.00) */
2  #include<stdio.h>
3  #include<stdlib.h>
4
5  struct node
6  {
7      int data;
8      struct node *next;
9  };
10 typedef struct node NODE_TYPE;
11
12 /* ----- */
13 void linked_list_print (NODE_TYPE *node)
14 {
15     while (NULL != node) {
16         printf ("%d-", node->data);
17         node = node->next;
18     }
19     printf ("\n");
20 }
21
22 /* ----- */
23 int main ()
24 {
25     NODE_TYPE *node;
26     node = malloc (sizeof (NODE_TYPE));
27     node->data = 300;
28     node->next = malloc (sizeof (NODE_TYPE));
29     node->next->data = 400;
30     node->next->next = malloc (sizeof (NODE_TYPE));
31     node->next->next->data = 500;
32     node->next->next->next = malloc (sizeof (NODE_TYPE));
33     node->next->next->next->data = 600;
34     node->next->next->next->next = NULL;
35     linked_list_print (node);

```

```
36     return 0;  
37 }
```

## ex13-2.c

```

1  /* code: ex13-2.c    (v1.20.00) */
2  #include<stdio.h>
3  #include<stdlib.h>
4
5  #define NOTFOUND (-1)
6  #define DATA_SIZE 6
7
8  struct node
9  {
10     int data;
11     struct node *next;
12 };
13 typedef struct node NODETYPE;
14
15 /* ----- */
16 void linked_list_insert_head (NODETYPE **head, int
    data)
17 {
18     NODETYPE *new_node;
19     new_node = malloc (sizeof (NODETYPE));
20     new_node->data = data;
21     if (*head == NULL) {
22         new_node->next = NULL;
23         *head = new_node;
24     }
25     else {
26         new_node->next = *head;
27         *head = new_node;
28     }
29 }
30
31 /* ----- */
32 void linked_list_print (NODETYPE *head)
33 {
34     printf ("Linked_list - [ -");

```

```
35     while (NULL != head) {
36         printf ("%02d-", head->data);
37         head = head->next;
38     }
39     printf ("]\n");
40 }
41
42 /* ----- */
43 int main ()
44 {
45     NODE_TYPE *head;
46     int i, data1;
47
48     head = NULL;
49     for (i = 0; i < DATA_SIZE; i++) {
50         data1 = (int) rand () % 100;
51         printf ("inserting-(head):-");
52         printf ("%02d\n", data1);
53         linked_list_insert_head (&head, data1);
54     }
55     linked_list_print (head);
56     return 0;
57 }
```

## ex13-3.c

```

1  /* code: ex13-3.c    (v1.20.00) */
2  #include<stdio.h>
3  #include<stdlib.h>
4
5  #define NOTFOUND (-1)
6  #define DATA_SIZE 6
7
8  struct node
9  {
10     int data;
11     struct node *next;
12 };
13 typedef struct node NODE_TYPE;
14
15 /* ----- */
16 int linked_list_delete_head (NODE_TYPE **head)
17 {
18     int data;
19     NODE_TYPE *temp;
20     if (*head == NULL) {
21         return NOTFOUND;
22     }
23     data = (*head)->data;
24     temp = (*head);
25     *head = (*head)->next;
26     free (temp);
27     return data;
28 }
29
30 /* ----- */
31 void linked_list_insert_head (NODE_TYPE **head, int
    data)
32 {
33     NODE_TYPE *new_node;
34     new_node = malloc (sizeof (NODE_TYPE));

```

```

35 new_node->data = data;
36 if (*head == NULL) {
37     new_node->next = NULL;
38     *head = new_node;
39 }
40 else {
41     new_node->next = *head;
42     *head = new_node;
43 }
44 }
45
46 /* ----- */
47 void linked_list_print (NODE_TYPE *head)
48 {
49     printf ("Linked list - [ -");
50     while (NULL != head) {
51         printf ("%02d -", head->data);
52         head = head->next;
53     }
54     printf (" ]\n");
55 }
56
57 /* ----- */
58 int main ()
59 {
60     NODE_TYPE *head;
61     int i, data1;
62
63     head = NULL;
64     for (i = 0; i < DATA_SIZE; i++) {
65         data1 = (int) rand () % 100;
66         printf ("inserting - (head): -");
67         printf ("%02d\n", data1);
68         linked_list_insert_head (&head, data1);
69     }
70     linked_list_print (head);
71     for (i = 0; i < DATA_SIZE / 2; i++) {
72         printf ("deleting - (head): -");

```

```
73     data1 = linked_list_delete_head (&head);  
74     printf ("%02d\n", data1);  
75 }  
76 linked_list_print (head);  
77 return 0;  
78 }
```

## ex13-4.c

```

1  /* code: ex13-4.c    (v1.20.00) */
2  #include<stdio.h>
3  #include<stdlib.h>
4
5  #define NOTFOUND (-1)
6  #define DATA_SIZE 6
7
8  struct node
9  {
10     int data;
11     struct node *next;
12 };
13 typedef struct node NODE_TYPE;
14
15 /* ----- */
16 int linked_list_search_node (NODE_TYPE *head, int key)
17 {
18     int i;
19     i = 0;
20     while (NULL != head) {
21         if (key == head->data) {
22             return i;
23         }
24         head = head->next;
25         i++;
26     }
27     return NOTFOUND;
28 }
29
30 /* ----- */
31 int linked_list_length (NODE_TYPE *head)
32 {
33     int c;
34     c = 0;
35     while (NULL != head) {

```



```

36     head = head->next;
37     c++;
38 }
39 return c;
40 }
41
42 /* ----- */
43 void linked_list_print (NODE_TYPE *head)
44 {
45     printf ("Linked list - [ -");
46     while (NULL != head) {
47         printf ("%02d -", head->data);
48         head = head->next;
49     }
50     printf (" ]\n");
51 }
52
53 /* ----- */
54 void linked_list_insert_head (NODE_TYPE **head, int
55     data)
56 {
57     NODE_TYPE *new_node;
58     new_node = malloc (sizeof (NODE_TYPE));
59     new_node->data = data;
60     if (*head == NULL) {
61         new_node->next = NULL;
62         *head = new_node;
63     }
64     else {
65         new_node->next = *head;
66         *head = new_node;
67     }
68 }
69 /* ----- */
70 int linked_list_delete_head (NODE_TYPE **head)
71 {
72     int data;

```

```

73     NODE_TYPE *temp;
74     if (*head == NULL) {
75         return NOT_FOUND;
76     }
77     data = (*head)->data;
78     temp = (*head);
79     *head = (*head)->next;
80     free (temp);
81     return data;
82 }
83
84 /* ----- */
85 void linked_list_insert_tail (NODE_TYPE **head, int
    data)
86 {
87     NODE_TYPE *new_node;
88     NODE_TYPE *temp;
89     new_node = malloc (sizeof (NODE_TYPE));
90
91     new_node->data = data;
92     new_node->next = NULL;
93     if (*head == NULL) {
94         *head = new_node;
95         temp = new_node;
96     }
97     else {
98         temp = *head;
99         while (temp->next != NULL) {
100             temp = temp->next;
101         }
102         temp->next = new_node;
103     }
104 }
105
106 /* ----- */
107 int linked_list_delete_tail (NODE_TYPE **head)
108 {
109     int data;

```

```

110     NODE_TYPE *temp;
111     NODE_TYPE *prev;
112
113     data = NOTFOUND;
114     if (*head == NULL) {
115         return data;
116     }
117     else {
118         temp = *head;
119         prev = *head;
120         while (temp->next != NULL) {
121             prev = temp;
122             temp = temp->next;
123         }
124         data = temp->data;
125         if ((*head)->next == NULL) {
126             *head = NULL;
127         }
128         else {
129             prev->next = NULL;
130         }
131         free (temp);
132     }
133     return data;
134 }
135
136
137 /* ----- */
138 void linked_list_delete_all (NODE_TYPE **head)
139 {
140     NODE_TYPE *current;
141     NODE_TYPE *next;
142
143     current = *head;
144     while (current != NULL) {
145         next = current->next;
146         free (current);
147         current = next;

```

```

148     }
149     *head = NULL;
150 }
151
152
153 /* ----- */
154 int main ()
155 {
156     NODE_TYPE *head;
157     int i, data1;
158
159     head = NULL;
160     for (i = 0; i < DATA_SIZE; i++) {
161         data1 = (int) rand () % 100;
162         printf ("adding-(head-node):-");
163         printf ("%02d\n", data1);
164         linked_list_insert_head (&head, data1);
165     }
166     linked_list_print (head);
167
168     for (i = 0; i < DATA_SIZE; i++) {
169         data1 = (int) rand () % 100;
170         printf ("adding-(tail-node):-");
171         printf ("%02d\n", data1);
172         linked_list_insert_tail (&head, data1);
173     }
174     linked_list_print (head);
175
176     for (i = 0; i < DATA_SIZE / 2; i++) {
177         printf ("deleting-(head-node):-");
178         data1 = linked_list_delete_head (&head);
179         printf ("%02d\n", data1);
180     }
181     linked_list_print (head);
182
183     for (i = 0; i < DATA_SIZE / 2; i++) {
184         printf ("deleting-(tail-node):-");
185         data1 = linked_list_delete_tail (&head);

```

```
186     printf ("%02d\n", data1);
187 }
188 linked_list_print (head);
189 printf ("Length of the linked list :%d\n",
        linked_list_length (head));
190
191 printf ("deleting entire linked list\n");
192 linked_list_delete_all (&head);
193 linked_list_print (head);
194 return 0;
195 }
```

# q13-1.c

```

1  /* code: q13-1.c    (v1.20.00) */
2  #include<stdio.h>
3  #include<stdlib.h>
4
5  #define NOTFOUND (-1)
6  #define DATA_SIZE 6
7
8  struct node
9  {
10     int data;
11     struct node *next;
12 };
13 typedef struct node NODE_TYPE;
14
15 /* ----- */
16 int linked_list_search_node (NODE_TYPE *head, int key)
17 {
18     int i;
19     i = 0;
20     while (NULL != head) {
21         if (key == head->data) {
22             return i;
23         }
24         head = head->next;
25         i++;
26     }
27     return NOTFOUND;
28 }
29
30 /* ----- */
31 void linked_list_print (NODE_TYPE *head)
32 {
33     printf ("Linked_list-[ ");
34     while (NULL != head) {
35         printf ("%02d-", head->data);

```

```

36     head = head->next;
37 }
38 printf ("]\n");
39 }
40
41
42 /* ----- */
43 void linked_list_insert_head (NODE_TYPE **head, int
    data)
44 {
45     NODE_TYPE *new_node;
46     new_node = malloc (sizeof (NODE_TYPE));
47     new_node->data = data;
48     if (*head == NULL) {
49         new_node->next = NULL;
50         *head = new_node;
51     }
52     else {
53         new_node->next = *head;
54         *head = new_node;
55     }
56 }
57
58 /* ----- */
59 int main ()
60 {
61     NODE_TYPE *head;
62     int i, data1, data2, stat;
63
64     head = NULL;
65     data2 = 0;
66     for (i = 0; i < DATA_SIZE; i++) {
67         data1 = (int) rand () % 100;
68         printf ("adding (head-node): ");
69         printf ("%02d\n", data1);
70         linked_list_insert_head (&head, data1);
71         if (i == (DATA_SIZE / 2)) {
72             data2 = data1;

```

```
73     }
74 }
75
76 linked_list_print (head);
77
78 stat = linked_list_search_node (head, data2);
79 if (stat == NOTFOUND) {
80     printf ("not found: %d", data2);
81 }
82 else {
83     printf ("found: %d - (%d)", data2, stat);
84 }
85 return 0;
86 }
```



## q13-2.c

```

1  /* code: q13-2.c    (v1.20.00) */
2  #include<stdio.h>
3  #include<stdlib.h>
4
5  #define NOTFOUND (-1)
6  #define DATA_SIZE 6
7
8  struct node
9  {
10     int data;
11     struct node *next;
12 };
13 typedef struct node NODETYPE;
14
15 /* ----- */
16 int linked_list_length (NODETYPE *head)
17 {
18     int c;
19     c = 0;
20     while (NULL != head) {
21         head = head->next;
22         c++;
23     }
24     return c;
25 }
26
27 /* ----- */
28 void linked_list_print (NODETYPE *head)
29 {
30     printf ("Linked list - [ -");
31     while (NULL != head) {
32         printf ("%02d -", head->data);
33         head = head->next;
34     }
35     printf ("]\n");

```

```

36 }
37
38
39 /* ----- */
40 void linked_list_insert_head (NODE_TYPE **head, int
    data)
41 {
42     NODE_TYPE *new_node;
43     new_node = malloc (sizeof (NODE_TYPE));
44     new_node->data = data;
45     if (*head == NULL) {
46         new_node->next = NULL;
47         *head = new_node;
48     }
49     else {
50         new_node->next = *head;
51         *head = new_node;
52     }
53 }
54
55 /* ----- */
56 int main ()
57 {
58     NODE_TYPE *head;
59     int i, data1;
60
61     head = NULL;
62     for (i = 0; i < DATA_SIZE; i++) {
63         data1 = (int) rand () % 100;
64         printf ("adding-(head-node):-");
65         printf ("%02d\n", data1);
66         linked_list_insert_head (&head, data1);
67     }
68     linked_list_print (head);
69     printf ("number-of-node(s):-%d\n", linked_list_length
        (head));
70
71     return 0;

```

72

{

---

---

## q13-3.c

```

1  /* code: q13-3.c    (v1.20.00) */
2  #include<stdio.h>
3  #include<stdlib.h>
4
5  #define NOTFOUND (-1)
6  #define DATA_SIZE 6
7
8  struct node
9  {
10     int data;
11     struct node *next;
12 };
13 typedef struct node NODE_TYPE;
14
15 /* ----- */
16 void linked_list_print (NODE_TYPE *head)
17 {
18     printf ("Linked list - [ -");
19     while (NULL != head) {
20         printf ("%02d-", head->data);
21         head = head->next;
22     }
23     printf (" ]\n\n");
24 }
25
26
27 /* ----- */
28 void linked_list_insert_tail (NODE_TYPE **head, int
    data)
29 {
30     NODE_TYPE *new_node;
31     NODE_TYPE *temp;
32     new_node = malloc (sizeof (NODE_TYPE));
33
34     new_node->data = data;

```

```
35 new_node->next = NULL;
36 if (*head == NULL) {
37     *head = new_node;
38     temp = new_node;
39 }
40 else {
41     temp = *head;
42     while (temp->next != NULL) {
43         temp = temp->next;
44     }
45     temp->next = new_node;
46 }
47 }
48
49 /* _____ */
50 int linked_list_delete_tail (NODE_TYPE **head)
51 {
52     int data;
53     NODE_TYPE *temp;
54     NODE_TYPE *prev;
55
56     data = NOTFOUND;
57     if (*head == NULL) {
58         return data;
59     }
60     else {
61         temp = *head;
62         prev = *head;
63         while (temp->next != NULL) {
64             prev = temp;
65             temp = temp->next;
66         }
67         data = temp->data;
68         if ((*head)->next == NULL) {
69             *head = NULL;
70         }
71         else {
72             prev->next = NULL;
```

```
73     }
74     free (temp);
75 }
76 return data;
77 }
78
79
80
81 /* ----- */
82 int main ()
83 {
84     NODE_TYPE *head;
85     int i, data1;
86
87     head = NULL;
88     for (i = 0; i < DATA_SIZE; i++) {
89         data1 = (int) rand () % 100;
90         printf ("adding-(tail-node):-");
91         printf ("%02d\n", data1);
92         linked_list_insert_tail (&head, data1);
93     }
94
95
96     linked_list_print (head);
97
98     for (i = 0; i < DATA_SIZE; i++) {
99         printf ("deleting-(tail-node):-");
100         data1 = linked_list_delete_tail (&head);
101         printf ("%02d\n", data1);
102     }
103
104     linked_list_print (head);
105     return 0;
106 }
```

## q13-4.c

```

1  /* code: q13-4.c    (v1.20.00) */
2  #include<stdio.h>
3  #include<stdlib.h>
4
5  #define DATA_SIZE 10
6
7  struct node
8  {
9      int data;
10     struct node *next;
11 };
12 typedef struct node NODE_TYPE;
13
14 struct linked_list
15 {
16     NODE_TYPE *head;
17     NODE_TYPE *tail;
18     NODE_TYPE *current;
19 };
20 typedef struct linked_list LINKED_LIST;
21
22 /* ----- */
23 void linked_list_init (LINKED_LIST *list)
24 {
25     list->head = NULL;
26     list->tail = NULL;
27     list->current = NULL;
28 }
29
30
31 /* ----- */
32 void linked_list_insert_node_h (LINKED_LIST *list, int
    data)
33 {
34     NODE_TYPE *node;

```

```
35
36     node = malloc (sizeof (NODETYPE));
37
38     node->data = data;
39
40     if (NULL == list->head) {
41         list->tail = node;
42         node->next = NULL;
43     }
44     else {
45         node->next = list->head;
46     }
47     list->head = node;
48 }
49
50 /* ----- */
51 void linked_list_insert_node_t (LINKED_LIST *list , int
    data)
52 {
53     NODETYPE *node;
54
55     node = malloc (sizeof (NODETYPE));
56
57     node->data = data;
58     node->next = NULL;
59
60     if (NULL == list->head) {
61         list->head = node;
62     }
63     else {
64         list->tail->next = node;
65     }
66     list->tail = node;
67 }
68
69 /* ----- */
70 NODETYPE *linked_list_find_node (LINKED_LIST *list ,
    int data)
```



```

71 | {
72 |     NODE_TYPE *node;
73 |
74 |     node = list->head;
75 |     while (NULL != node) {
76 |         if (node->data == data) {
77 |             return node;
78 |         }
79 |         node = node->next;
80 |     }
81 |     return NULL;
82 | }
83 |
84 |
85 | /* _____ */
86 | void linked_list_delete_node (LINKED_LIST *list ,
87 |     NODE_TYPE *node)
88 | {
89 |     if (node == list->head) {
90 |         if (NULL == list->head->next) {
91 |             list->head = NULL;
92 |             list->tail = NULL;
93 |         }
94 |         else {
95 |             list->head = list->head->next;
96 |         }
97 |     }
98 |     else {
99 |         NODE_TYPE *temp;
100 |         temp = list->head;
101 |         while ((NULL != temp) && (temp->next != node)) {
102 |             temp = temp->next;
103 |         }
104 |         if (NULL != temp) {
105 |             temp->next = node->next;
106 |         }
107 |     }
108 |     free (node);

```

```
108 }
109
110 /* ----- */
111 void linked_list_print (LINKED_LIST *list)
112 {
113     NODE_TYPE *node;
114
115     printf ("linked-list - [ -");
116     node = list->head;
117     while (NULL != node) {
118         printf ("%02d -", node->data);
119         node = node->next;
120     }
121     printf (" ]\n");
122 }
123
124 /* ----- */
125 int linked_list_count_node (LINKED_LIST *list)
126 {
127     NODE_TYPE *node;
128     int i;
129
130     i = 0;
131     node = list->head;
132     while (NULL != node) {
133         i++;
134         node = node->next;
135     }
136     return i;
137 }
138
139
140
141 /* ----- */
142 int main ()
143 {
144     LINKED_LIST *list;
145     NODE_TYPE *node;
```

```

146     int i, data1, data2, del_data;
147
148     list = malloc (sizeof (LINKED_LIST));
149     linked_list_init (list);
150
151     for (i = 0; i < DATA_SIZE; i++) {
152         data1 = (int) rand () % 100;
153         printf ("adding-node-to-head-of-linked-list:-");
154         printf ("%02d\n", data1);
155         linked_list_insert_node_h (list, data1);
156     }
157     linked_list_print (list);
158
159
160     for (i = 0; i < DATA_SIZE; i++) {
161         data2 = (int) rand () % 100;
162         printf ("adding-node-to-tail-of-linked-list:-");
163         printf ("%02d\n", data2);
164         linked_list_insert_node_t (list, data2);
165     }
166     linked_list_print (list);
167
168     del_data = data2;
169     printf ("finding-node:-[%d]\n", del_data);
170     node = linked_list_find_node (list, del_data);
171     if (NULL != node) {
172         printf ("deleting-node:-[%d]\n", del_data);
173         linked_list_delete_node (list, node);
174     }
175     else {
176         printf ("Node-not-found:-[%d]\n", del_data);
177     }
178     linked_list_print (list);
179
180     printf ("number-of-node(s):%d\n",
181            linked_list_count_node (list));
182     free (list);

```

```
183 |  
184 |   return 0;  
185 | }
```

## ex14-1.c

```

1  /* code: ex14-1.c    (v1.20.00) */
2  #include<stdio.h>
3  #include<stdlib.h>
4
5  #define STACK_UNDERFLOW (-1)
6  #define DATA_SIZE 6
7
8  struct node
9  {
10     int data;
11     struct node *next;
12 };
13 typedef struct node NODE_TYPE;
14
15
16 /* ----- */
17 void stack_push (NODE_TYPE **head, int data)
18 {
19     NODE_TYPE *new_node;
20     new_node = malloc (sizeof (NODE_TYPE));
21     new_node->data = data;
22     new_node->next = *head;
23     *head = new_node;
24 }
25
26 /* ----- */
27 int stack_pop (NODE_TYPE **head)
28 {
29     int data;
30     NODE_TYPE *temp;
31     if (*head == NULL) {
32         return STACK_UNDERFLOW;
33     }
34     data = (*head)->data;
35     temp = (*head);

```

```

36     *head = (*head)->next;
37     free (temp);
38     return data;
39 }
40
41 /* ----- */
42 void stack_print (NODE_TYPE *head)
43 {
44     if (head == NULL) {
45         printf ("stack is empty.\n");
46         return;
47     }
48     printf ("stack-[");
49     while (NULL != head) {
50         printf ("%02d-", head->data);
51         head = head->next;
52     }
53     printf ("]\n");
54 }
55
56 /* ----- */
57 int main ()
58 {
59     NODE_TYPE *stack;
60     int i, data1;
61     stack = NULL;
62     for (i = 0; i < DATA_SIZE; i++) {
63         data1 = (int) rand () % 100;
64         printf ("push:-");
65         printf ("%02d\n", data1);
66         stack_push (&stack, data1);
67     }
68     stack_print (stack);
69     for (i = 0; i < DATA_SIZE / 2; i++) {
70         printf ("pop:-");
71         data1 = stack_pop (&stack);
72         printf ("%02d\n", data1);
73     }

```

```
74 |   stack_print (stack);  
75 |   return 0;  
76 | }
```

## ex14-2.c

```

1  /* code: ex14-2.c    (v1.20.00) */
2  #include <stdio.h>
3  #include <string.h>
4  #include <stdlib.h>
5
6  #define DATA_SIZE 6
7  #define QUEUEEMPTY (-1)
8
9  struct node
10 {
11     int data;
12     struct node *next;
13 };
14 typedef struct node NODE_TYPE;
15
16 /* ----- */
17 void q_enqueue (NODE_TYPE **front, NODE_TYPE **rear, int
18     data)
19 {
20     NODE_TYPE *new_node;
21     new_node = malloc (sizeof (NODE_TYPE));
22     new_node->data = data;
23     new_node->next = NULL;
24     if (*rear == NULL) {
25         *front = *rear = new_node;
26     }
27     else {
28         (*rear)->next = new_node;
29         *rear = new_node;
30     }
31 }
32 /* ----- */
33 int q_dequeue (NODE_TYPE **front, NODE_TYPE **rear)
34 {

```



```

35     int data;
36     NODE_TYPE *temp;
37     if (*front == NULL) {
38         return QUEUEEMPTY;
39     }
40     temp = *front;
41     data = (*front)->data;
42     if (*front == *rear) {
43         *front = *rear = NULL;
44     }
45     else {
46         *front = (*front)->next;
47     }
48     free (temp);
49     return data;
50 }
51
52 /* ----- */
53 void q_print (NODE_TYPE *front)
54 {
55     printf ("queue-[-");
56     while (front != NULL) {
57         printf ("%02d-", front->data);
58         front = front->next;
59     }
60     printf ("]\n");
61 }
62
63 /* ----- */
64 int main ()
65 {
66     int i, data1;
67     NODE_TYPE *front, *rear;
68
69     front = NULL;
70     rear = NULL;
71     for (i = 0; i < DATA_SIZE; i++) {
72         data1 = (int) rand () % 100;

```

```
73     printf ("enqueue:-");
74     printf ("%02d\n", data1);
75     q_enqueue (&front , &rear , data1);
76 }
77 q_print (front);
78 for (i = 0; i < DATA_SIZE / 2; i++) {
79     printf ("dequeue:-");
80     data1 = q_dequeue (&front , &rear);
81     printf ("%02d\n", data1);
82 }
83 q_print (front);
84
85 return 0;
86 }
```

## q14-1.c

```

1  /* code: q14-1.c    (v1.20.00) */
2
3  #include<stdio.h>
4  #include<stdlib.h>
5
6  #define DATA_SIZE 6
7
8  /* doubly linked list */
9  struct node
10 {
11     int data;
12     struct node *prev;
13     struct node *next;
14 };
15 typedef struct node NODE_TYPE;
16
17 /* ----- */
18 void dll_print_head (NODE_TYPE *head)
19 {
20     NODE_TYPE *temp;
21
22     temp = head;
23     if (temp == NULL) {
24         printf ("List is empty\n");
25         return;
26     }
27     printf ("print -(head):-");
28     while (temp->next != NULL) {
29         printf ("%d-", temp->data);
30         temp = temp->next;
31     }
32     printf ("%d-\n", temp->data);
33 }
34
35

```

```

36  /* ----- */
37  void dll_print_tail (NODETYPE *tail)
38  {
39      NODETYPE *temp;
40
41      temp = tail;
42      if (temp == NULL) {
43          printf ("List is empty\n");
44          return;
45      }
46      printf ("print --(tail):-");
47      while (temp->prev != NULL) {
48          printf ("%d-", temp->data);
49          temp = temp->prev;
50      }
51      printf ("%d-\n", temp->data);
52  }
53
54
55  /* ----- */
56  void dll_insert_head (NODETYPE **head, NODETYPE **
57      tail, int data)
58  {
59      NODETYPE *new_node;
60
61      new_node = malloc (sizeof (NODETYPE));
62      new_node->data = data;
63      new_node->prev = NULL;
64      new_node->next = NULL;
65
66      if (*head == NULL) {
67          *head = new_node;
68          *tail = *head;
69      }
70      else {
71          new_node->next = *head;
72          (*head)->prev = new_node;
73          *head = new_node;

```

```

73     }
74 }
75
76
77 /* ----- */
78 void dll_insert_tail (NODE_TYPE **head, NODE_TYPE **
    tail, int data)
79 {
80     NODE_TYPE *new_node;
81
82     new_node = malloc (sizeof (NODE_TYPE));
83     new_node->data = data;
84     new_node->prev = NULL;
85     new_node->next = NULL;
86
87     if (*head == NULL) {
88         *head = new_node;
89         new_node = *head;
90     }
91     else {
92         (*tail)->next = new_node;
93         new_node->prev = *tail;
94         *tail = new_node;
95     }
96 }
97
98
99 /* ----- */
100 int main ()
101 {
102     NODE_TYPE *head;
103     NODE_TYPE *tail;
104     int i, data;
105
106     head = NULL;
107     tail = NULL;
108     for (i = 0; i < DATA_SIZE; i++) {
109         data = (int) rand () % 100;

```

```
110     printf ("adding-(head):-");
111     printf ("%02d\n", data);
112     dll_insert_head (&head, &tail, data);
113 }
114 dll_print_head (head);
115 dll_print_tail (tail);
116
117 for (i = 0; i < DATA_SIZE; i++) {
118     data = (int) rand () % 100;
119     printf ("adding-(tail):-");
120     printf ("%02d\n", data);
121     dll_insert_tail (&head, &tail, data);
122 }
123 dll_print_head (head);
124 dll_print_tail (tail);
125
126 return 0;
127 }
```

## q14-2.c

```

1  /* code: q14-2.c    (v1.20.00) */
2  #include<stdio.h>
3  #include<stdlib.h>
4
5  #define DATA_SIZE 6
6
7  /* circular doubly linked list */
8  struct node
9  {
10     int data;
11     struct node *prev;
12     struct node *next;
13 };
14 typedef struct node NODE_TYPE;
15
16 /* ----- */
17 void cdll_print_head (NODE_TYPE *head, NODE_TYPE *tail)
18 {
19
20     if ((head == tail) && (head == NULL)) {
21         printf ("List is empty\n");
22         return;
23     }
24     printf ("print --(head):-");
25     while (head->next != tail->next) {
26         printf ("%d-", head->data);
27         head = head->next;
28     }
29     printf ("%d-\n", head->data);
30 }
31
32
33 /* ----- */
34 void cdll_print_tail (NODE_TYPE *head, NODE_TYPE *tail)
35 {

```

```

36
37     if ((head == tail) && (head == NULL)) {
38         printf ("List is empty\n");
39         return;
40     }
41     printf ("print --(tail):-");
42     while (head->prev != tail->prev) {
43         printf ("%d-", tail->data);
44         tail = tail->prev;
45     }
46     printf ("%d-\n", tail->data);
47 }
48
49
50 /* ----- */
51 void cdll_insert_head (NODE_TYPE **head, NODE_TYPE **
    tail, int data)
52 {
53     NODE_TYPE *new_node;
54
55     new_node = malloc (sizeof (NODE_TYPE));
56     new_node->data = data;
57     new_node->prev = NULL;
58     new_node->next = NULL;
59
60     if ((*head == *tail) && (*head == NULL)) {
61         *head = new_node;
62         *tail = new_node;
63         (*head)->prev = NULL;
64         (*head)->next = NULL;
65         (*tail)->prev = NULL;
66         (*tail)->next = NULL;
67     }
68     else {
69         new_node->next = *head;
70         (*head)->prev = new_node;
71         *head = new_node;
72         (*head)->prev = *tail;

```



```

73     (*tail)->next = *head;
74 }
75 }
76
77
78
79
80 /* ----- */
81 void cdll_insert_tail (NODE_TYPE **head, NODE_TYPE **
    tail, int data)
82 {
83     NODE_TYPE *new_node;
84
85     new_node = malloc (sizeof (NODE_TYPE));
86     new_node->data = data;
87     new_node->prev = NULL;
88     new_node->next = NULL;
89
90     if ((*head == *tail) && (*head == NULL)) {
91         *head = new_node;
92         *tail = new_node;
93         (*head)->prev = NULL;
94         (*head)->next = NULL;
95         (*tail)->prev = NULL;
96         (*tail)->next = NULL;
97     }
98     else {
99         (*tail)->next = new_node;
100         new_node->prev = *tail;
101         *tail = new_node;
102         (*head)->prev = *tail;
103         (*tail)->next = *head;
104     }
105 }
106
107
108 /* ----- */
109 int main ()

```

```
110 {
111     NODE_TYPE *head;
112     NODE_TYPE *tail;
113     int i, data;
114
115     head = NULL;
116     tail = NULL;
117     for (i = 0; i < DATA_SIZE; i++) {
118         data = (int) rand () % 100;
119         printf ("adding-(head):-");
120         printf ("%02d\n", data);
121         cdll_insert_head (&head, &tail, data);
122     }
123     cdll_print_head (head, tail);
124     cdll_print_tail (head, tail);
125
126     for (i = 0; i < DATA_SIZE; i++) {
127         data = (int) rand () % 100;
128         printf ("adding-(tail):-");
129         printf ("%02d\n", data);
130         cdll_insert_tail (&head, &tail, data);
131     }
132     cdll_print_head (head, tail);
133     cdll_print_tail (head, tail);
134
135     return 0;
136 }
```