1. 구간 $[1,6]$에서 정의된 함수 $f(x) = x^2 - 16$를 생각해 보자. 오차한계 ϵ=1e-5가 되도록 이분법과 뉴턴법을 이용하여 근사적인 해를 구하라. 각 알고리즘에 대하여 k (반복수), x_k (근사적인 해), 그리고 참값과의 차이인 $|x_k - 4|$를 출력하고 주어진 오차한계를 만족시키는지 확인하라.

```
Bisection = function(x0, x1, epsilon = 1e-5)
{
  fx0 = f(x0)
  fx1 = f(x1)
  if (fx0 * fx1 >0)
    return("wrong initial values")
  error = abs(x1 - x0)
  N = 1
  #에러가 입실론보다 작아지면 stop
  while (error > epsilon)
  {
    N = N + 1
    error = error / 2
    x2 = (x0 + x1) / 2
    fx2 = f(x2)
    #곱해서 0이 되면 그 안에 해가 있다고 가정
    if (fx0 * fx2 < 0)
    {
      x1 = x2; fx1 = fx2
    } else
    {
      x0 = x2; fx0 = fx2
    }
  }

  return(list(x = x2, n = N))
}
```

```
> f = function(x) {x^2-16}
> bisection_result = Bisection(1,6)
> bisection_result
$x
[1] 4.000002

$n
[1] 20

> bisection_result$x-4
[1] 1.907349e-06
```

```
Newton = function(x0, epsilon = 1e-5, n = 100)
{
  e = 1
  N = 1
  d = epsilon
  while (e > epsilon)
  {
    N = N + 1
    if (N > n)
      return("not converge after 100 iterations")
    x1 = x0 - f(x0) * d / (f(x0 + d) - f(x0))
    e = abs(x1 - x0)
    x0 = x1
  }

  return(list(x = x1, n = N))
}
```

```
> Newton(1)
$x
[1] 4

$n
[1] 7

> Newton(6)
$x
[1] 4

$n
[1] 6

> #1에서 시작했을 경우는 7번, 6에서 시작했을 경우는 6번만에 수렴한다.
> Newton(1)$x-4
[1] 8.508749e-13
> Newton(6)$x-4
[1] 0
```

2. $\int_0^1 \exp(-x)dx = 1 - e^{-1}$이다. 직사각형법, 사다리꼴법, 심슨법을 사용하여 적분값을 구하라. 각 알고리즘에 대하여 n (구간수), I_n (적분값), 그리고 참값과의 차이 $|I_n - (1 - e^{-1})|$을 출력하라. 편의상 전체 구간수는 $n = 2, 4, \ldots, 20$까지만 해보도록 한다.

```
Integral = function(a, b, n)
{
  integral = 0
  h = (b - a) / n
  for (i in 1:n)
    integral = integral + h * f(a + (i-1/2) * h)

  return(integral)
}
```

```r
Trapezoid = function(a, b, n = 50)
{
  h = (b - a) / n
  integral = (f(a) + f(b)) / 2

  x = a
  n1 = n - 1
  for (i in 1:n1)
  {
    x = x + h
    integral = integral + f(x)
  }
  integral = integral * h

  return(integral)
}
```

```r
Simpson = function(a, b, n = 12)
{
  h = (b - a) / n
  integral = f(a) + f(b)
  x2 = a
  x3 = a + h
  even = 0
  odd = f(x3)
  h2 = 2 * h
  n1 = n / 2 - 1
  for (i in 1:n1)
  {
    x2 = x2 + h2
    x3 = x3 + h2
    even = even + f(x2)
    odd = odd + f(x3)
  }
  integral = (integral + 4 * odd + 2 * even) * h / 3

  return(integral)
}
```

```
> f = function(x) {exp(-x)}
> integral.answer<-data.frame() ;trapezoid.answer<-data.frame() ;simpson.answer<-data.frame()
> for (i in 1:10){
+   integral.answer<-rbind(integral.answer, data.frame(2*i, Integral(0, 1, 2*i), abs(Integral(0, 1, 2*i) - (1-exp(-1)))))
+   trapezoid.answer<-rbind(trapezoid.answer, data.frame(2*i, Trapezoid(0, 1, 2*i), abs(Trapezoid(0, 1, 2*i) - (1-exp(-1)))))
+   simpson.answer<-rbind(simpson.answer, data.frame(2*i, Simpson(0, 1, 2*i), abs(Simpson(0, 1, 2*i) - (1-exp(-1)))))
+ }
> colnames(integral.answer)<-c('n', '적분값', '오차')
> colnames(trapezoid.answer)<-c('n', '적분값', '오차')
> colnames(simpson.answer)<-c('n', '적분값', '오차')
> #직사각형법의 적분 오차
> integral.answer
    n    적분값         오차
1   2 0.6255837 6.536891e-03
2   4 0.6304774 1.643151e-03
3   6 0.6313895 7.310287e-04
4   8 0.6317092 4.113494e-04
5  10 0.6318573 2.633068e-04
6  12 0.6319377 1.828682e-04
7  14 0.6319862 1.343594e-04
8  16 0.6320177 1.028725e-04
9  18 0.6320393 8.128391e-05
10 20 0.6320547 6.584109e-05
> #사다리꼴법의 적분 오차
> trapezoid.answer
    n    적분값         오차
1   2 0.6452352 0.0131146313
2   4 0.6354094 0.0032888702
3   6 0.6335831 0.0014625651
4   8 0.6329434 0.0008228594
5  10 0.6326472 0.0005266794
6  12 0.6324863 0.0003657682
7  14 0.6323893 0.0002687359
8  16 0.6323263 0.0002057550
9  18 0.6322831 0.0001625741
10 20 0.6322522 0.0001316863
```

```
> #심슨적분법의 적분 오차
> simpson.answer
    n    적분값         오차
1   2 1.0035487 3.714281e-01
2   4 0.6321342 1.361649e-05
3   6 0.6321233 2.700773e-06
4   8 0.6321214 8.557762e-07
5  10 0.6321209 3.507605e-07
6  12 0.6321207 1.692168e-07
7  14 0.6321207 9.135904e-08
8  16 0.6321206 5.356062e-08
9  18 0.6321206 3.344089e-08
10 20 0.6321206 2.194210e-08
```

3. 제약조건 $x_1 - 2x_2 \le 9, 3x_2 + x_3 \le 9, x_2 + x_4 \le 10$하에서 $C(x) = x_1 + 3x_2 + 4x_3 + x_4$를 최대화하는 $x_1, x_2, x_3, x_4 \ge 0$의 해는 무엇인가?

$\max(x_1 + 3x_2 + 4x_3 + x_4)$ subject to $x_1 - 2x_2 \le 9,\ 3x_2 + x_3 \le 9\ \ x_2 + x_4 \le 10$ (단, $x_1, x_2, x_3, x_4\ \ge 0$)

$$\begin{pmatrix} 1 & -2 & 0 & 0 \\ 0 & 3 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \le \begin{pmatrix} 9 \\ 9 \\ 10 \end{pmatrix}$$

```
> eg.lp = lp(objective.in=c(1,3,4,1), #objective.in : 계수
+           const.mat=matrix(c(1,-2,0,0,0,3,1,0,0,1,0,1),nrow=3, byrow = TRUE),
+           const.rhs=c(9,9,10),
+           const.dir=c("<=","<=", '<='), direction="max")
> eg.lp
Success: the objective function is 55
> eg.lp$solution
[1]  9  0  9 10
```

따라서, $x_1 = 9, x_2 = 0, x_3 = 9, x_4 = 10$이다.

4. 예제 R 코드의 이차계획법에 의한 portfolio 최적화 예제에서 한 주식에 40%이상 투자할 수 없다고 하자. 이 경우에 최적의 투자 전략은 무엇인가?

$$\min\left(\frac{1}{2}\beta^T \begin{pmatrix} 0.01 & 0.002 & 0.002 \\ 0.002 & 0.01 & 0.002 \\ 0.002 & 0.002 & 0.01 \end{pmatrix} \beta - \begin{pmatrix} 0.002 \\ 0.005 \\ 0.01 \end{pmatrix}^T \beta\right) \text{ subject to } \begin{pmatrix} 1 & 1 & 0 & 0 & -1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & -1 \end{pmatrix}^T \beta \ge \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ -0.4 \\ -0.4 \\ -0.4 \end{pmatrix}$$

```
> A = cbind(rep(1,3), diag(rep(1,3)), -diag(rep(1,3)))
> D = matrix(c(.01,.002,.002,.002,.01,.002,.002,.002,.01), nrow=3)
> x = c(.002,.005,.01)
> b = c(1,0,0,0, -0.4, -0.4, -0.4)
> qp = solve.QP(2*D, x, A, b, meq = 1)
> qp$solution
[1] 0.20625 0.39375 0.40000
```

따라서, 최적의 투자 전략은 주식 1에 20.625%, 주식 2에 39.375%, 주식 3에 40%를 투자하는 것이다.