

2022-1 (인)컴퓨팅사고와 파이썬 프로그래밍

기말고사 문제풀이



2022. 6. 11.

교수 김 영 탁

영남대학교 정보통신공학과

(Tel : +82-53-810-2497; E-mail : ytkim@yu.ac.kr)

Exam2A

◆ Exam2A. (40점, 시험 시간 (09:00 ~ 09:45) 45분)

(1) 요구사항

- Excel 파일 (Exam2A_student_scores.xlsx)에 학생 5명의 국어, 영어, 수학, 과학 성적을 표로 준비하라.
- 이 Excel 파일을 pandas의 read_excel() 함수를 사용하여 읽고, 데이터 프레임을 생성하라.
- 각 학생들의 성적 평균을 'Avg' 열 (column)을 추가하라.
- 데이터 프레임을 학생 성적 평균을 기준으로 내림차순으로 정렬하라.
- 데이터 프레임의 과목별 평균을 계산하여 "Per_class_Avg" 행 (row)를 추가하라.
- 종합정리된 데이터 프레임을 화면으로 출력하라.
- 종합정리된 데이터 프레임을 Excel file (Exam2A_processed_scores.xlsx)로 출력하라.
- 프로그램 출력 첫 부분에 "2022-1 컴사파 Exam2A 학번: 00000000, 성명: 홍길동" 양식으로 본인 학번과 이름을 출력할 것.



(2) 입력 데이터 파일 - Exam2A_scores.xlsx

	A	B	C	D	E	F
1	st_id	st_name	Kor	Eng	Math	Sci
2	21234	Hong	95.7	92.3	95.2	75.9
3	22021	Kim	92.4	94.5	93.5	92.4
4	21203	Park	85.7	88.7	90.3	87.3
5	21057	Choi	98.9	97.2	98.2	95.3
6	22512	Lee	80.2	95.7	75.9	91.3

(3) main() 함수 – 예시

```
.... # 기본 주석문
# pandas 모듈 import
# 핵심 기능 구현
# - 입력 데이터 파일로부터 데이터 입력 및 pandas 데이터 프레임 (df) 생성, 출력
# - 학생별 평균 성적 계산 및 avgs_per_student 데이터 프레임 생성
# - 데이터 프레임 df에 avgs_per_student를 'Avg' 레이블과 함께 추가
# - 데이터 프레임 df에서 각 과목별 평균성적 계산 및 avgs_per_class에 저장
# - 데이터 프레임 df를 Avg 열 기준의 내림차순으로 정렬하고, df_sorted 데이터 프레임에 대입
# - 데이터 프레임 df_sorted에 avgs_per_class를 마지막 행으로 추가
# - 마지막 행 (avgs_per_class)의 'st_name' 원소를 'Per_class_Avg'로 설정
# - 데이터 프레임 df_sorted를 Exam2A_processed_scores.xlsx에 저장
```

(4) 실행 결과 – 화면 출력 (예시) - df_with_avg, df_sorted_with_avg

```
df_with_avg =
  st_id st_name  Kor  Eng  Math  Sci  Avg
0  21234   Hong  95.7  92.3  95.2  75.9  89.775
1  22021   Kim  92.4  94.5  93.5  92.4  93.200
2  21203   Park  85.7  88.7  90.3  87.3  88.000
3  21057   Choi  98.9  97.2  98.2  95.3  97.400
4  22512   Lee  80.2  95.7  75.9  91.3  85.775

df_sorted_with_avg =
  st_id st_name  Kor  Eng  Math  Sci  Avg
3  21057.0   Choi  98.90  97.20  98.20  95.30  97.400
1  22021.0    Kim  92.40  94.50  93.50  92.40  93.200
0  21234.0   Hong  95.70  92.30  95.20  75.90  89.775
2  21203.0   Park  85.70  88.70  90.30  87.30  88.000
4  22512.0   Lee  80.20  95.70  75.90  91.30  85.775
5           Per_class_Avg  90.58  93.68  90.62  88.44  90.830
```

(5) 실행 결과 – Exam2A_processed_scores.xlsx

	A	B	C	D	E	F	G	H
1		st_id	st_name	Kor	Eng	Math	Sci	Avg
2	3	21057	Choi	98.9	97.2	98.2	95.3	97.4
3	1	22021	Kim	92.4	94.5	93.5	92.4	93.2
4	0	21234	Hong	95.7	92.3	95.2	75.9	89.775
5	2	21203	Park	85.7	88.7	90.3	87.3	88
6	4	22512	Lee	80.2	95.7	75.9	91.3	85.775
7	5		Per_class_Avg	90.58	93.68	90.62	88.44	90.83



(6) 결과물 제출

- 파이썬 소스코드 : Exam2A_학번_성명.py
- Excel 입력 데이터 파일 : Exam2A_student_scores.xlsx
- Excel 출력 데이터 파일 : Exam2A_processed_scores.xlsx
- 실행 결과 (capture된 이미지) (Exam2A실행결과_학번_성명.png)

(7) 본인 응시 및 사용 컴퓨터 IP 주소 확인을 위한 동영상 파일 제출 !!



```
# Exam2A - pandas - df, calculation of average of each class, save to Excel
# <기본 주석문>
# Author :
# Date :
# Brief description :

import pandas as pd

df = pd.read_excel("Exam2A_student_scores.xlsx")
print("df =\n", df)

df_tmp = df.loc[:, ['Kor', 'Eng', 'Math', 'Sci']]
print("\ndf_tmp =\n", df_tmp)

avgs_per_student = df_tmp.mean(1) # mean with axes 1
print("\navgs_per_student =\n", avgs_per_student)

df.loc[:, 'Avg'] = avgs_per_student
print("\ndf_with_avg =\n", df)

df_tmp = df.loc[:, ['Kor', 'Eng', 'Math', 'Sci', 'Avg']]
avgs_per_class = df_tmp.mean() # mean with axes 0
print("\navgs_per_class =\n", avgs_per_class)

df_sorted = df.sort_values(by='Avg', ascending=False)

df_sorted.loc[len(df_sorted), ['Kor', 'Eng', 'Math', 'Sci', 'Avg']] = avgs_per_class
df_sorted.at[len(df_sorted)-1, 'st_id'] = ''
df_sorted.at[len(df_sorted)-1, 'st_name'] = 'Per_class_Avg'
print("\ndf_sorted_with_avg =\n", df_sorted)

print("Writing df to excel file")
with pd.ExcelWriter("Exam2A_processed_scores.xlsx") as excel_writer:
    df_sorted.to_excel(excel_writer, sheet_name='Students Records')
```

```
df =
   st_id st_name  Kor  Eng  Math  Sci
0  21234   Hong  95.7  92.3  95.2  75.9
1  22021   Kim   92.4  94.5  93.5  92.4
2  21203   Park  85.7  88.7  90.3  87.3
3  21057   Choi  98.9  97.2  98.2  95.3
4  22512   Lee   80.2  95.7  75.9  91.3

df_tmp =
      Kor  Eng  Math  Sci
0  95.7  92.3  95.2  75.9
1  92.4  94.5  93.5  92.4
2  85.7  88.7  90.3  87.3
3  98.9  97.2  98.2  95.3
4  80.2  95.7  75.9  91.3

avgs_per_student =
0    89.775
1    93.200
2    88.000
3    97.400
4    85.775
dtype: float64

df_with_avg =
   st_id st_name  Kor  Eng  Math  Sci  Avg
0  21234   Hong  95.7  92.3  95.2  75.9  89.775
1  22021   Kim   92.4  94.5  93.5  92.4  93.200
2  21203   Park  85.7  88.7  90.3  87.3  88.000
3  21057   Choi  98.9  97.2  98.2  95.3  97.400
4  22512   Lee   80.2  95.7  75.9  91.3  85.775

avgs_per_class =
Kor    90.58
Eng    93.68
Math   90.62
Sci    88.44
Avg    90.83
dtype: float64

df_sorted_with_avg =
   st_id st_name  Kor  Eng  Math  Sci  Avg
3  21057.0   Choi  98.90  97.20  98.20  95.30  97.400
1  22021.0   Kim   92.40  94.50  93.50  92.40  93.200
0  21234.0   Hong  95.70  92.30  95.20  75.90  89.775
2  21203.0   Park  85.70  88.70  90.30  87.30  88.000
4  22512.0   Lee   80.20  95.70  75.90  91.30  85.775
5          Per_class_Avg  90.58  93.68  90.62  88.44  90.830
Writing df to excel file
```



Exam2B

◆ Exam2B. (35점, 시험 시간 (09:45 ~ 10:30) 45분)

(1) 요구사항

- 환율에 따라 미국 달러와 한국 원화를 계산하는 환전 계산기 구현을 위하여 class USD_KRW_Calculator()를 구현하라.
- class USD_KRW_Calculator()의 __init__() 함수에는 tkinter LabelFrame 생성 및 배치, tkinter GUI label 및 entry 생성 및 배치, 버튼 생성 및 배치 기능을 구현하라. LabelFrame의 title은 "USD_KRW Exchange Calculator"으로 설정하고, GROOVE 모양의 테두리를 사용하라.
- 이 환전 계산기에는 1 미국 달러에 대한 한국 원화 (Korean Won)의 환율을 입력하는 Entry, 미국 달러 금액을 입력/출력하는 Entry, 한국 원화 금액을 입력/출력 entry가 있으며, 각 항목에 대한 label이 표시된다.
- 각 금액의 출력은 오른쪽 맞춤으로 정렬할 것 (아래 GUI 입출력 예시 참조)
- 환율에 따른 미국 달러 -> 한국 원화 계산을 위하여 "US Dollar -> Kr Won" 녹색 버튼을 구현하고, 환율에 따른 한국 원화 -> 미국 달러 계산을 위하여 "Kr Won -> US Dollar" 노란색 버튼을 구현하라. 버튼은 아래 GUI 예시와 같이 한 줄에 배치하라.
- 환율, 미국 달러, 한국 원화의 입력/출력은 실수형 (float 또는 double)으로 표시하며, 소숫점 이하 2자리까지 정리하여 출력하라.
- 생성된 tkinter GUI 윈도우 제목에 "2022-1 컴사파 Exam2B 학번: 00000000, 성명: 홍길동" 양식으로 본인 학번과 이름을 출력할 것.



(2) GUI 입출력 (예시)

2022-1 컴사파 Exam2B 학번 이름

USD_KRW Exchange Calculator

Exchange Ratio (1 USD => x KRW)	1245.23
US Dollar	250
Korean Won	311307.5

US Dollar -> Kr Won Kr Won -> US Dollar

2022-1 컴사파 Exam2B 학번 이름

USD_KRW Exchange Calculator

Exchange Ratio (1 USD => x KRW)	1245.23
US Dollar	80.31
Korean Won	100000

US Dollar -> Kr Won Kr Won -> US Dollar

(3) main() 함수 (예시)

```
.... # 기본 주석문
.... # 필요한 모듈 import
class USD_KRW_Calculator(master):
def __init__(self, master):
.... # 이 부분은 직접 구현할 것
def convert_USD_KRW(self):
.... # 이 부분은 직접 구현할 것
def convert_KRW_USD(self):
.... # 이 부분은 직접 구현할 것
def main():
    win = Tk()
    .... # win의 title로 2022-1 Exam2B 학번 이름 설정
    app = USD_KRW_Calculator(win)
    win.mainloop()
if __name__ == "__main__":
    main()
```

(4) 결과물 제출

- 파이썬 소스코드 (Exam2B_학번_성명.py)
- 실행 결과 (capture된 이미지) (Exam2B실행결과_학번_성명.png)

```

# Exam2B tkinter GUI, Calc_Student_Scores (1)
# <기본 주석문>
# Author :
# Date :
# Brief description :

from math import *
from tkinter import *

class USD_KRW_Calculator():
    def __init__(self, master):
        frame = LabelFrame(master, text="USD_KRW Exchange Calculator", relief=GROOVE)
        frame.pack()
        self.USD_KRW_Ratio = DoubleVar()
        Label(frame, text = 'Exchange Ratio (1 USD => x KRW)').grid(row=0, column=0)
        Entry(frame, textvariable=self.USD_KRW_Ratio, justify=RIGHT).grid(row=0, column=1)

        self.us_dollar_var = DoubleVar()
        Label(frame, text='US Dollar').grid(row=1, column=0)
        Entry(frame, textvariable=self.us_dollar_var, justify=RIGHT).grid(row=1, column=1)

        self.kr_won_var = DoubleVar()
        Label(frame, text='Korean Won').grid(row=2, column=0)
        Entry(frame, textvariable=self.kr_won_var, justify=RIGHT).grid(row=2, column=1)

        button = Button(frame, text='US Dollar -> Kr Won', command=self.convert_USD_KRW, bg="green")
        button.grid(row=3, column=0)
        button = Button(frame, text='Kr Won -> US Dollar', command=self.convert_KRW_USD, bg="yellow")
        button.grid(row=3, column=1)

```

```
# Exam2B tkinter GUI, Calc_Student_Scores (2)
```

```
def convert_USD_KRW(self):
```

```
    currency_ratio = self.USD_KRW_Ratio.get()
    usd = self.us_dollar_var.get()
    krw = usd * currency_ratio
    krw_round = round(krw, 2)
    self.kr_won_var.set(krw_round)
```

```
def convert_USD_KRW(self):
```

```
    currency_ratio = self.USD_KRW_Ratio.get()
    usd = self.us_dollar_var.get()
    krw = usd * currency_ratio
    krw_round = round(krw, 2)
    self.kr_won_var.set(krw_round)
```

```
def convert_KRW_USD(self):
```

```
    currency_ratio = self.USD_KRW_Ratio.get()
    krw = self.kr_won_var.get()
    usd = krw / currency_ratio
    usd_round = round(usd, 2)
    self.us_dollar_var.set(usd_round)
```

```
def main():
```

```
    global window
    win = Tk()
    win.wm_title('2022-1 컴사파 Exam2B 학번 이름')
    app = USD_KRW_Calculator(win)
    win.mainloop()
```

```
if __name__ == "__main__":
    main()
```



Exam2C

◆ Exam2C. (35점, 시험 시간 (10:30 ~ 11:15) 45분)

(1) 요구사항

- 다음 선형 방정식으로 구성된 선형시스템 $A \cdot X = B$ 의 해 (a, b, c, d, e 의 값으로 구성된 배열 X)를 NumPy 확장 모듈을 사용하여 구하는 파이썬 프로그램을 작성하라.

$$\begin{aligned} a + 5b + 3c + 3d + 7e &= 105 \\ 3a + 4b + 5c + 6d + 7e &= 135 \\ a + 3b + 5c + 7d + 9e &= 145 \\ 3a + b + 4c + d + 5e &= 74 \\ 5a + 5b + 3c + 3d + e &= 75 \end{aligned}$$

- Numpy 배열 A와 B를 구성하고, 각각 출력하라.
- 배열 A의 행렬식 (determinant) \det_A 를 구하여 출력하라.
- 배열 A의 역행렬 (inverse matrix) inv_A 를 구하여 출력하라.
- Numpy 확장 모듈의 $\text{solve}()$ 함수를 사용하여 선형시스템의 해 X 를 산출하고, 이를 출력하라.
- Numpy 확장 모듈의 $\text{matmul}()$ 함수를 사용하여 행렬 곱셈 $B1 = A \cdot X$ 를 계산하여 출력하고, 이를 행렬 B와 비교하라.
- Numpy 확장 모듈의 $\text{matmul}()$ 함수를 사용하여 행렬 곱셈 $X1 = \text{inv_A} \cdot B$ 를 계산하여 출력하고, 이를 행렬 X와 비교하라.
- 프로그램 출력 첫 부분에 "2022-1 컴사파 Exam2C 학번: 00000000, 성명: 홍길동" 양식으로 본인 학번과 이름을 출력할 것.

(2) main() 함수 예시

```
..... # 기본 주석문
..... # numpy의 import
# 핵심 기능 구현
# 선형시스템  $AX = B$ 에서 A와 B의 준비
# - numpy 배열 A의 생성 및 출력
# - numpy 배열 B의 생성 및 출력
# - 배열 A의 행렬식 (det_A) 계산 및 출력
# - 배열 A의 역행렬 (inv_A) 계산 및 출력
# - 선형시스템  $AX = B$ 의 해 (solution)인 X의 산출 및 출력
# -  $B1 = A * X$ 의 계산 및 B1 출력
# -  $X1 = inv\_A * B$ 의 계산 및 X1 출력
```

(3) IDLE shell 출력

2022-1 컴사파 Exam2C 학번: 0000, 이름: 홍길동

```
A =  
[ ]  
[ ]  
[ ]  
[ ]  
B =  
[ ]  
det_A =  
-20.000000000000156  
inv_A =  
[[ 5.00000000e-01  2.46000000e+01 -1.68000000e+01 -3.00000000e+00  
   -9.50000000e+00]  
 [ 1.84297022e-15 -8.80000000e+00  5.90000000e+00  1.00000000e+00  
   3.50000000e+00]  
 [-1.00000000e+00 -3.84000000e+01  2.62000000e+01  5.00000000e+00  
   1.50000000e+01]  
 [-1.17385776e-15  6.60000000e+00 -4.30000000e+00 -1.00000000e+00  
   -2.50000000e+00]  
 [ 5.00000000e-01  1.64000000e+01 -1.12000000e+01 -2.00000000e+00  
   -6.50000000e+00]]  
X =  
[ ]  
B1 = A * X =  
[ ]  
X1 = inv_A * B =  
[ ]
```

(4) 결과물 제출

- 파이썬 소스코드 (Exam2C_학번_성명.py)
- 실행 결과 (capture된 이미지) (Exam2C실행결과_실행결과_학번_성명.png)

```

# Exam2C - Numpy-based Linear System Solution
# <기본 주석문>
# Author :
# Date :
# Brief description :

import numpy as np

print("2022-1 컴사파 Exam2C 학번: 0000, 이름: 홍길동")
A = np.array([[1, 5, 3, 3, 7], [3, 4, 5, 6, 7], [1, 3, 5, 7, 9], [3, 1, 4, 1, 5], [5, 5, 3, 3, 1]])
B = np.array([105, 135, 145, 74, 75])
print("A =\n", A)
print("B = \n", B)
det_A = np.linalg.det(A)
print("det_A =\n", det_A)
inv_A = np.linalg.inv(A)
print("inv_A =\n", inv_A)
X = np.linalg.solve(A, B)
print("X = \n", X)
B1 = np.matmul(A, X)
print("B1 = A * X = \n", B1)
X1 = np.matmul(inv_A, B)
print("X1 = inv_A * B =\n", X1)

```



Exam2D

◆ Exam2D. (40점, 시험 시간 (11:15 ~ 12:00) 45분)

(1) 요구사항

- 난수의 개수 (N)이 주어질 때, $-N/200 \sim +(N-1)/200$ 범위의 중복되지 않는 실수자료형 (float) 난수 리스트를 생성하여 반환하는 함수 `genRandFloatList(N)`를 사용자 정의 모듈 `MyList.py`에 구현하라.
- 실수자료형 리스트의 첫 부분과 끝 부분의 리스트 원소를 한 줄 당 `per_line` 원소 x `sample_lines` 줄 출력하는 함수 `printFloatListSample(L, per_line, sample_lines)`를 `MyList.py`에 구현하라. float 자료형인 리스트 원소는 소숫점 이하 2 자리 까지 출력할 것.
- 주어진 float 자료형 리스트를 오름차순으로 병합 정렬하는 함수 `mergeSort(L)`을 사용자 정의 모듈 `MySortings.py`에 구현하라.
- 주어진 float 자료형 리스트를 오름차순으로 퀵정렬하는 함수 `quickSort(L)`을 사용자 정의 모듈 `MySortings.py`에 구현하라.
- `main()` 함수에서는 난수 리스트의 크기 (`L_size`)가 포함된 리스트를 사용하여 전체 기능 시험이 반복구조로 실행되도록 하라.
- `main()` 함수에서 `genRandFloatList()` 함수를 사용하여 중복되지 않는 난수 리스트를 생성한 후, `printFloatListSampe()` 함수를 사용하여 출력하며, `mergeSort()` 함수를 사용하여 정렬한 후, 정렬된 리스트를 출력하라.
- `main()` 함수에는 정렬된 리스트를 `random` 모듈의 `shuffle()` 함수를 사용하여 뒤섞은 후, 출력하고, `quickSort()` 함수를 사용하여 오름차순으로 정렬한 후, 출력하라.
- `mergeSort()`와 `quickSort()`를 수행할 때 걸린 경과시간을 측정하여 초 단위로 출력하라.
- 프로그램 출력 첫 부분에 "2022-1 검사파 Exam2D 학번: 00000000, 성명: 홍길동" 양식으로 본인 학번과 이름을 출력할 것.



(2) main() 함수 예시

```
# Exam2d - Performance Comparisons of Sorting Algorithms (quickSort, mergeSort)
.... # 필요한 모듈 import
def main():
    .... # 2022-1 컴사파 Exam2D 학번: 00000000, 성명: 홍길동 출력
    L = [100000, 500000, 1000000, 5000000]
    for L_size in L:
        print("\nGenerating random list of size ({}).format(L_size))
        L = MyList.genRandFloatList(L_size)
        # testing MergeSorting
        print("Before mergeSort of L :")
        MyList.printFloatListSample(L, 10, 3)
        t1 = time.time()
        L = MySortings.mergeSort(L)
        t2 = time.time()
        print("\nAfter mergeSort of L :")
        MyList.printFloatListSample(L, 10, 3)
        time_elapsed = t2 - t1
        print("Merge sorting of list (size={}) took {} sec".format(L_size, time_elapsed))

    # testing Quick Sorting
    random.shuffle(L)
    print("Before quickSort of L :")
    MyList.printFloatListSample(L, 10, 3)
    t1 = time.time()
    MySortings.quickSort(L)
    t2 = time.time()
    print("After quickSort of L :")
    MyList.printFloatListSample(L, 10, 3)
    time_elapsed = t2 - t1
    print("Quick sorting of list (size={}) took {} sec".format(L_size, time_elapsed))

if __name__ == "__main__":
    main()
```



(3) shell 입출력 예시

2022-1 원서과 기말고사 학번: 0000, 이름: 홍길동

```
Generating random list of size (100000) ...
Before mergeSort of L :
173.74 -36.61 -203.82 221.40 86.70 -23.76 471.33 146.47 330.72 -440.90
339.66 735.35 42.90 306.59 -407.63 -421.48 -343.66 329.73 57.05 -150.90
46.08 170.32 15.06 426.36 370.96 320.77 483.73 419.50 307.74 -261.62
...
275.81 -152.00 303.92 30.38 -391.04 -249.12 -457.98 -204.62 -393.87 -149.85
-430.13 163.85 -105.61 10.17 -233.17 395.43 -56.40 418.69 388.85 -307.11
17.33 -91.55 400.41 492.66 301.07 437.89 -496.85 27.72 257.90 305.11

After mergeSort of L :
-500.00 -499.99 -499.98 -499.97 -499.96 -499.95 -499.94 -499.93 -499.92 -499.91
-499.90 -499.89 -499.88 -499.87 -499.86 -499.85 -499.84 -499.83 -499.82 -499.81
-499.80 -499.79 -499.78 -499.77 -499.76 -499.75 -499.74 -499.73 -499.72 -499.71
...
499.70 499.71 499.72 499.73 499.74 499.75 499.76 499.77 499.78 499.79
499.80 499.81 499.82 499.83 499.84 499.85 499.86 499.87 499.88 499.89
499.90 499.91 499.92 499.93 499.94 499.95 499.96 499.97 499.98 499.99

Merge sorting of list (size=100000) took 0.23106029593262 sec
Before quickSort of L :
-138.68 -248.41 -30.56 432.42 -69.34 -138.83 188.41 -325.44 34.57 -236.86
45.00 48.63 -140.47 -343.47 -361.35 -314.47 -66.79 -52.58 400.70 -135.04
316.37 -407.39 230.46 -399.85 -43.52 -63.18 -109.91 41.71 423.75
...
-409.99 305.08 -379.48 -245.44 303.34 217.95 7.44 -154.07 -24.96 408.39
87.09 -54.61 -315.85 89.73 116.22 -311.85 294.32 -447.31 -420.16 -481.14
-365.44 177.57 42.99 -225.27 94.29 -456.70 334.78 -91.57 -365.30 -179.90

After quickSort of L :
-500.00 -499.99 -499.98 -499.97 -499.96 -499.95 -499.94 -499.93 -499.92 -499.91
-499.90 -499.89 -499.88 -499.87 -499.86 -499.85 -499.84 -499.83 -499.82 -499.81
-499.80 -499.79 -499.78 -499.77 -499.76 -499.75 -499.74 -499.73 -499.72 -499.71
...
499.70 499.71 499.72 499.73 499.74 499.75 499.76 499.77 499.78 499.79
499.80 499.81 499.82 499.83 499.84 499.85 499.86 499.87 499.88 499.89
499.90 499.91 499.92 499.93 499.94 499.95 499.96 499.97 499.98 499.99

Quick sorting of list (size=100000) took 0.182542340061821 sec
```

```
Generating random list of size (100000) ...
Before mergeSort of L :
149.06 -995.22 4030.43 59.90 2908.36 2307.63 2726.07 -301.17 -4093.81 1007.99
1881.90 4551.06 3207.11 879.59 -4663.93 -3508.50 3206.62 -301.15 1362.07 4620.56
3594.92 -2304.40 4791.20 3006.04 4171.23 -4071.58 -2620.43 -777.72 -603.34 -2990.44
...
4473.37 1154.37 3034.99 -2175.56 4091.34 -4357.66 -3105.65 -1316.14 -3002.88 1209.61
1305.92 -2121.61 2291.00 2949.63 -3630.15 -2560.20 4093.66 934.59 -3751.31 -4021.02
2361.17 109.53 -1818.64 -4736.09 205.92 4625.04 -2942.40 -646.19 -4795.92 -3399.73

After mergeSort of L :
-5000.00 -4999.99 -4999.98 -4999.97 -4999.96 -4999.95 -4999.94 -4999.93 -4999.92 -4999.91
-4999.90 -4999.89 -4999.88 -4999.87 -4999.86 -4999.85 -4999.84 -4999.83 -4999.82 -4999.81
-4999.80 -4999.79 -4999.78 -4999.77 -4999.76 -4999.75 -4999.74 -4999.73 -4999.72 -4999.71
...
4999.70 4999.71 4999.72 4999.73 4999.74 4999.75 4999.76 4999.77 4999.78 4999.79
4999.80 4999.81 4999.82 4999.83 4999.84 4999.85 4999.86 4999.87 4999.88 4999.89
4999.90 4999.91 4999.92 4999.93 4999.94 4999.95 4999.96 4999.97 4999.98 4999.99

Merge sorting of list (size=100000) took 2.909179630279541 sec
Before quickSort of L :
4513.95 3046.25 3833.44 4189.26 -1168.06 225.23 -2676.79 2733.16 3205.27 -1560.49
-1290.23 -396.63 -205.04 707.42 -2247.62 3016.59 -4342.20 2003.55 544.66 -1040.07
-4001.74 -2027.52 -055.23 1727.62 -1526.31 4122.81 3426.40 3706.72 3074.53 4747.68
...
-2084.12 -2166.73 3623.30 -2009.31 682.42 4493.17 1516.56 -4686.31 2624.77 3304.68
3514.40 4084.87 -2192.29 -2635.19 -303.41 -3091.37 3052.52 -1056.30 1416.74 3406.93
-1838.24 -1204.64 -1604.48 2168.70 2082.50 2331.87 -504.56 -4704.71 3702.43 4700.62

After quickSort of L :
-5000.00 -4999.99 -4999.98 -4999.97 -4999.96 -4999.95 -4999.94 -4999.93 -4999.92 -4999.91
-4999.90 -4999.89 -4999.88 -4999.87 -4999.86 -4999.85 -4999.84 -4999.83 -4999.82 -4999.81
-4999.80 -4999.79 -4999.78 -4999.77 -4999.76 -4999.75 -4999.74 -4999.73 -4999.72 -4999.71
...
4999.70 4999.71 4999.72 4999.73 4999.74 4999.75 4999.76 4999.77 4999.78 4999.79
4999.80 4999.81 4999.82 4999.83 4999.84 4999.85 4999.86 4999.87 4999.88 4999.89
4999.90 4999.91 4999.92 4999.93 4999.94 4999.95 4999.96 4999.97 4999.98 4999.99

Quick sorting of list (size=100000) took 2.321063200105951 sec
```

```
Generating random list of size (500000) ...
Before mergeSort of L :
1461.40 -1045.23 1559.76 -1462.09 -1922.73 -2001.00 2190.68 -405.72 -147.18 1239.73
1632.93 1301.93 997.26 -20.75 293.18 -1077.04 1701.14 -295.38 -2272.32 322.26
-2355.00 1448.34 -47.92 -1234.31 994.67 2012.45 1326.99 1242.35 1537.96 1504.83
...
-930.49 1263.05 -1863.69 681.75 -530.64 1216.43 -1025.20 -2058.83 2308.68 1075.98
-610.51 -574.08 441.55 644.11 -1100.06 60.77 -2337.33 92.27 513.76 1960.90
659.30 2222.36 -1339.84 -1658.99 -2129.64 1348.52 682.91 -2096.59 -700.36 1316.06

After mergeSort of L :
-2500.00 -2499.99 -2499.98 -2499.97 -2499.96 -2499.95 -2499.94 -2499.93 -2499.92 -2499.91
-2499.90 -2499.89 -2499.88 -2499.87 -2499.86 -2499.85 -2499.84 -2499.83 -2499.82 -2499.81
-2499.80 -2499.79 -2499.78 -2499.77 -2499.76 -2499.75 -2499.74 -2499.73 -2499.72 -2499.71
...
2499.70 2499.71 2499.72 2499.73 2499.74 2499.75 2499.76 2499.77 2499.78 2499.79
2499.80 2499.81 2499.82 2499.83 2499.84 2499.85 2499.86 2499.87 2499.88 2499.89
2499.90 2499.91 2499.92 2499.93 2499.94 2499.95 2499.96 2499.97 2499.98 2499.99

Merge sorting of list (size=500000) took 1.360950026263428 sec
Before quickSort of L :
1002.91 -1247.62 1614.77 -1452.32 -2017.14 921.70 971.58 500.47 -1232.19 -1203.17
-1752.47 -039.60 -1337.66 -059.77 1505.12 -1048.82 617.55 -531.29 -1948.14 -223.18
-2495.51 -2206.66 1032.27 -1323.58 -1511.80 -446.30 1193.40 003.08 1365.55 -1395.40
...
-1476.23 -423.66 -2379.32 -2176.28 -34.60 2014.53 2326.22 700.14 1308.26 -1527.08
-2156.43 -612.29 -1843.09 -1575.03 001.40 -094.65 -2441.77 -023.59 -1444.09 -70.54
-099.42 -518.70 -2028.78 -1045.00 -30.12 -3204.16 2120.50 -620.95 2332.78 -1539.08

After quickSort of L :
-2500.00 -2499.99 -2499.98 -2499.97 -2499.96 -2499.95 -2499.94 -2499.93 -2499.92 -2499.91
-2499.90 -2499.89 -2499.88 -2499.87 -2499.86 -2499.85 -2499.84 -2499.83 -2499.82 -2499.81
-2499.80 -2499.79 -2499.78 -2499.77 -2499.76 -2499.75 -2499.74 -2499.73 -2499.72 -2499.71
...
2499.70 2499.71 2499.72 2499.73 2499.74 2499.75 2499.76 2499.77 2499.78 2499.79
2499.80 2499.81 2499.82 2499.83 2499.84 2499.85 2499.86 2499.87 2499.88 2499.89
2499.90 2499.91 2499.92 2499.93 2499.94 2499.95 2499.96 2499.97 2499.98 2499.99

Quick sorting of list (size=500000) took 1.05524545423096 sec
```

```
Generating random list of size (500000) ...
Before mergeSort of L :
-4473.18 -1661.35 5705.67 9406.40 24103.20 -51120.17 4390.29 -13034.50 -257.10 5088.44
15343.51 -0100.10 791.13 13001.08 21182.70 13007.61 -13051.74 24622.38 16100.62 20011.83
-13081.95 -13003.19 -17079.95 -9032.97 -8341.01 929.07 10959.97 12240.92 -24277.22 15018.81
...
-6470.71 366.24 20824.08 -21369.90 -24279.43 12005.53 20042.11 -5101.17 22450.14 5061.40
20099.75 -13400.07 820.07 6723.95 -16830.26 14737.96 17996.13 -5047.38 -14409.89 17771.08
-8355.77 22093.48 6231.78 -10809.10 8148.06 21775.15 1150.87 16948.05 15278.83 16354.31

After mergeSort of L :
-25000.00 -24999.99 -24999.98 -24999.97 -24999.96 -24999.95 -24999.94 -24999.93 -24999.92 -24999.91
-24999.90 -24999.89 -24999.88 -24999.87 -24999.86 -24999.85 -24999.84 -24999.83 -24999.82 -24999.81
-24999.80 -24999.79 -24999.78 -24999.77 -24999.76 -24999.75 -24999.74 -24999.73 -24999.72 -24999.71
...
24999.70 24999.71 24999.72 24999.73 24999.74 24999.75 24999.76 24999.77 24999.78 24999.79
24999.80 24999.81 24999.82 24999.83 24999.84 24999.85 24999.86 24999.87 24999.88 24999.89
24999.90 24999.91 24999.92 24999.93 24999.94 24999.95 24999.96 24999.97 24999.98 24999.99

Merge sorting of list (size=500000) took 17.912758701004297 sec
Before quickSort of L :
-10387.94 -14927.20 -9801.65 -14079.26 11427.00 10122.67 20129.59 20948.48 19547.51 -17548.10
-21649.35 -17319.09 -15114.39 10854.64 -4952.62 3190.00 -16145.95 1917.16 -0479.07 -20160.70
19427.81 -9309.20 13353.40 -11604.51 10514.71 3620.67 2249.24 -24772.94 15049.19 -12176.55
...
-17771.30 19747.06 -10306.06 71.40 9336.30 -5013.31 -17072.08 14210.32 -20081.91 23035.08
-22275.42 -19317.21 235.64 15566.40 14164.75 -14602.41 4038.49 -21166.08 -5048.42 15426.15
-21414.55 7212.38 0713.20 -10237.02 16096.13 -0751.17 19355.54 -20573.27 15495.77 -17280.15

After quickSort of L :
-25000.00 -24999.99 -24999.98 -24999.97 -24999.96 -24999.95 -24999.94 -24999.93 -24999.92 -24999.91
-24999.90 -24999.89 -24999.88 -24999.87 -24999.86 -24999.85 -24999.84 -24999.83 -24999.82 -24999.81
-24999.80 -24999.79 -24999.78 -24999.77 -24999.76 -24999.75 -24999.74 -24999.73 -24999.72 -24999.71
...
24999.70 24999.71 24999.72 24999.73 24999.74 24999.75 24999.76 24999.77 24999.78 24999.79
24999.80 24999.81 24999.82 24999.83 24999.84 24999.85 24999.86 24999.87 24999.88 24999.89
24999.90 24999.91 24999.92 24999.93 24999.94 24999.95 24999.96 24999.97 24999.98 24999.99

Quick sorting of list (size=500000) took 14.410092350507573 sec
```

(4) 결과물 제출

- 사용자 정의 모듈 (MyList.py, MySortings.py), 파이썬 소스코드 (Exam2D_학번_성명.py)
- 실행 결과 (capture된 이미지) (Exam2D실행결과_학번_성명.png)



```
# Exam2d - Performance Comparisons of Sorting Algorithms (quickSort, mergeSort) (1)
```

```
# <기본 주석문>
```

```
# Author :
```

```
# Date :
```

```
# Brief description :
```

```
import os, sys, random, time
```

```
import MyList, MySortings
```

```
def main():
```

```
    print("2022-1 컴사파 기말고사 학번: 0000, 이름: 홍길동")
```

```
    L = [100000, 500000, 1000000, 5000000]
```

```
    for L_size in L:
```

```
        print("\nGenerating random list of size ({})." .format(L_size))
```

```
        L = MyList.genRandFloatList(L_size)
```

```
        # testing MergeSorting
```

```
        print("Before mergeSort of L :")
```

```
        MyList.printFloatListSample(L, 10, 3)
```

```
        t1 = time.time()
```

```
        L = MySortings.mergeSort(L)
```

```
        t2 = time.time()
```

```
        print("\nAfter mergeSort of L :")
```

```
        MyList.printFloatListSample(L, 10, 3)
```

```
        time_elapsed = t2 - t1
```

```
        print("Merge sorting of list (size={}) took {} sec" .format(L_size, time_elapsed))
```

```
        # testing Quick Sorting
```

```
        random.shuffle(L)
```

```
        print("Before quickSort of L :")
```

```
        MyList.printFloatListSample(L, 10, 3)
```

```
        t1 = time.time()
```

```
        MySortings.quickSort(L)
```

```
        t2 = time.time()
```

```
        print("After quickSort of L :")
```

```
        MyList.printFloatListSample(L, 10, 3)
```

```
        time_elapsed = t2 - t1
```

```
        print("Quick sorting of list (size={}) took {} sec" .format(L_size, time_elapsed))
```

```
if __name__ == "__main__":
```

```
    main()
```

```
# MyList.py (1)
# <기본 주석문>
# Author :
# Date :
# Brief description :
```

```
import random
```

```
def genRandFloatList(size):
    L = []
    offset = - size / 200
    for i in range(size):
        d = (i / 100 ) + offset
        L.append(d)
    random.shuffle(L)
    return L
```

```
# MyList.py (2)
```

```
def printFloatListSample(L, per_line = 10, sample_lines = 3):
    count = 0
    size = len(L)
    for i in range(0, sample_lines):
        s = ""
        for j in range(0, per_line):
            if count > size:
                break
            s += "{0:>10.2f} ".format(L[count])
            count += 1
        print(s)
        if count >= size:
            break
    if count < size:
        print(' . . . . .')
        if count < (size - per_line * sample_lines):
            count = size - per_line * sample_lines
        for i in range(0, sample_lines):
            s = ""
            for j in range(0, per_line):
                if count >= size:
                    break
                s += "{0:>10.2f} ".format(L[count])
                count += 1
            print(s)
            if count >= size:
                break
```



```

# MySorting.py (1)
# <기본 주석문>
# Author :
# Date :
# Brief description :

# mergeSort()
def mergeSort(arr):
    def _merge(arr_left, arr_right):
        arr_result = []
        i, j = 0, 0
        while i < len(arr_left) and j < len(arr_right):
            if arr_left[i] < arr_right[j]:
                arr_result.append(arr_left[i])
                i += 1
            else:
                arr_result.append(arr_right[j])
                j += 1
        while (i < len(arr_left)):
            arr_result.append(arr_left[i])
            i += 1
        while (j < len(arr_right)):
            arr_result.append(arr_right[j])
            j += 1
        return arr_result
    if len(arr) < 2:
        return arr[:]
    else:
        middle = len(arr) // 2
        left = mergeSort(arr[:middle])
        right = mergeSort(arr[middle:])
        return _merge(left, right)

```

```

# MySorting.py (2)

# quickSort()
def quickSort(arr):
    def _partition(arr, left, right, pivot):
        pv = arr[pivot]
        arr[pivot], arr[right] = arr[right], arr[pivot]
        np = i = left # np: new_pivot_index
        while (i <= right-1):
            if (arr[i] <= pv):
                arr[np], arr[i] = arr[i], arr[np]
                np += 1
            i += 1
        arr[np], arr[right] = arr[right], arr[np]
        return np

    def _quickSortLoop(arr, left, right):
        #print("quickSort", left, right)
        if (left >= right):
            return
        pi = (left + right) // 2
        new_pi = _partition(arr, left, right, pi)
        #print("after partition : ", left, right, new_pi)
        if (left < new_pi - 1):
            _quickSortLoop(arr, left, new_pi-1)
        if (new_pi + 1 < right):
            _quickSortLoop(arr, new_pi+1, right)
        size = len(arr)
        _quickSortLoop(arr, 0, size-1)

```

2022-1 컴사파 기말고사 학번: 0000, 이름: 홍길동

Generating random list of size (100000) ...

Before mergeSort of L :

-135.56	132.99	376.16	-187.64	207.17	482.27	-361.96	-187.12	452.25	499.07
-479.45	405.69	-449.81	102.35	-115.65	-362.71	-149.56	-350.09	112.67	354.74
341.79	-196.17	149.85	-26.62	-109.73	67.60	187.70	325.16	376.09	499.76
.									
-71.18	271.10	396.99	-10.41	-466.55	131.24	-461.05	196.41	-310.24	-80.51
-248.53	365.96	-420.06	453.42	480.74	-194.74	-365.28	334.59	-352.13	-325.79
-90.55	448.50	322.73	345.39	291.52	-427.87	142.89	192.30	-347.78	-486.93

After mergeSort of L :

-500.00	-499.99	-499.98	-499.97	-499.96	-499.95	-499.94	-499.93	-499.92	-499.91
-499.90	-499.89	-499.88	-499.87	-499.86	-499.85	-499.84	-499.83	-499.82	-499.81
-499.80	-499.79	-499.78	-499.77	-499.76	-499.75	-499.74	-499.73	-499.72	-499.71
.									
499.70	499.71	499.72	499.73	499.74	499.75	499.76	499.77	499.78	499.79
499.80	499.81	499.82	499.83	499.84	499.85	499.86	499.87	499.88	499.89
499.90	499.91	499.92	499.93	499.94	499.95	499.96	499.97	499.98	499.99

Merge sorting of list (size=100000) took 0.2254199981689453 sec

Before quickSort of L :

366.78	467.63	117.12	192.04	347.79	332.06	421.29	-486.45	-471.51	-360.91
-251.19	-406.70	147.36	-1.06	271.02	127.04	-124.42	243.64	-362.25	-248.92
-60.16	-187.99	-369.00	-313.57	437.73	-111.90	163.07	-9.95	141.81	133.81
.									
477.71	-234.37	10.73	285.29	462.39	-400.47	-44.88	30.54	291.41	387.22
-291.08	114.91	-383.12	-41.12	-178.42	165.21	-101.69	-478.67	-112.50	-2.90
-52.63	16.62	-301.33	-413.50	-225.23	93.48	227.06	64.24	-6.76	310.67

After quickSort of L :

-500.00	-499.99	-499.98	-499.97	-499.96	-499.95	-499.94	-499.93	-499.92	-499.91
-499.90	-499.89	-499.88	-499.87	-499.86	-499.85	-499.84	-499.83	-499.82	-499.81
-499.80	-499.79	-499.78	-499.77	-499.76	-499.75	-499.74	-499.73	-499.72	-499.71
.									
499.70	499.71	499.72	499.73	499.74	499.75	499.76	499.77	499.78	499.79
499.80	499.81	499.82	499.83	499.84	499.85	499.86	499.87	499.88	499.89
499.90	499.91	499.92	499.93	499.94	499.95	499.96	499.97	499.98	499.99

Quick sorting of list (size=100000) took 0.1715843677520752 sec



Generating random list of size (500000) ...

Before mergeSort of L :

-1376.39	-707.87	-71.78	-1881.20	-135.42	-2101.80	-807.83	-1609.80	634.05	1132.91
1887.59	755.90	-1449.80	2184.08	-1462.14	557.47	-1590.46	2288.29	-697.28	-1557.91
-1479.39	1695.12	-2204.85	2200.82	-2109.75	-1430.70	-523.17	-390.76	1333.97	-395.07
.									
-1056.51	339.74	2063.48	-2222.57	-954.12	2449.75	539.81	-1812.57	-853.79	-1828.61
1773.10	-1459.87	-2085.11	-433.81	1909.29	-2044.38	1457.04	-232.50	-1651.25	-1741.19
1981.71	-2351.45	749.60	947.90	2393.34	-447.04	-2487.96	-913.86	-1467.71	-766.59

After mergeSort of L :

-2500.00	-2499.99	-2499.98	-2499.97	-2499.96	-2499.95	-2499.94	-2499.93	-2499.92	-2499.91
-2499.90	-2499.89	-2499.88	-2499.87	-2499.86	-2499.85	-2499.84	-2499.83	-2499.82	-2499.81
-2499.80	-2499.79	-2499.78	-2499.77	-2499.76	-2499.75	-2499.74	-2499.73	-2499.72	-2499.71
.									
2499.70	2499.71	2499.72	2499.73	2499.74	2499.75	2499.76	2499.77	2499.78	2499.79
2499.80	2499.81	2499.82	2499.83	2499.84	2499.85	2499.86	2499.87	2499.88	2499.89
2499.90	2499.91	2499.92	2499.93	2499.94	2499.95	2499.96	2499.97	2499.98	2499.99

Merge sorting of list (size=500000) took 1.3194940090179443 sec

Before quickSort of L :

-2053.81	900.54	2221.71	-838.04	86.09	1453.16	-1274.12	1347.99	-612.21	1209.85
1542.14	1864.19	-1470.13	1826.31	2443.44	37.59	990.88	2012.77	-2003.35	-43.89
474.55	1883.70	-2443.04	-312.24	1234.48	-1157.36	-1841.39	-2464.68	1846.16	-138.29
.									
-516.93	-2264.94	-2102.80	-1171.82	1843.51	622.89	-711.99	1961.31	2496.77	1148.25
-643.42	-1135.03	2411.95	591.97	-335.16	-1563.36	-613.16	-41.05	-864.07	-1529.32
1524.28	-55.84	782.18	741.50	-1025.51	231.54	717.16	-1065.99	-1139.93	-2251.48

After quickSort of L :

-2500.00	-2499.99	-2499.98	-2499.97	-2499.96	-2499.95	-2499.94	-2499.93	-2499.92	-2499.91
-2499.90	-2499.89	-2499.88	-2499.87	-2499.86	-2499.85	-2499.84	-2499.83	-2499.82	-2499.81
-2499.80	-2499.79	-2499.78	-2499.77	-2499.76	-2499.75	-2499.74	-2499.73	-2499.72	-2499.71
.									
2499.70	2499.71	2499.72	2499.73	2499.74	2499.75	2499.76	2499.77	2499.78	2499.79
2499.80	2499.81	2499.82	2499.83	2499.84	2499.85	2499.86	2499.87	2499.88	2499.89
2499.90	2499.91	2499.92	2499.93	2499.94	2499.95	2499.96	2499.97	2499.98	2499.99

Quick sorting of list (size=500000) took 1.0132591724395752 sec



Generating random list of size (1000000) ...

Before mergeSort of L :

2574.25	-4305.15	2393.93	-1105.14	-2845.08	4740.57	2658.70	-2854.85	339.32	-4402.28
-4841.18	1170.94	98.92	-4425.43	2921.36	-2283.79	3885.08	-1180.41	-3071.97	1380.41
-338.87	4635.99	4923.23	-680.86	-4286.07	1634.92	1016.55	773.60	2895.11	-3719.32
.									
-3864.14	-4220.79	1105.96	-3241.59	4577.36	-1795.93	-315.51	1095.93	-389.16	-14.03
3195.83	-3073.58	-3317.10	-929.62	3304.06	1557.89	1893.89	2980.16	4804.26	2209.50
-4569.25	3363.88	-4177.78	454.21	-1898.31	912.81	-4462.03	788.51	-1946.11	1004.81

After mergeSort of L :

-5000.00	-4999.99	-4999.98	-4999.97	-4999.96	-4999.95	-4999.94	-4999.93	-4999.92	-4999.91
-4999.90	-4999.89	-4999.88	-4999.87	-4999.86	-4999.85	-4999.84	-4999.83	-4999.82	-4999.81
-4999.80	-4999.79	-4999.78	-4999.77	-4999.76	-4999.75	-4999.74	-4999.73	-4999.72	-4999.71
.									
4999.70	4999.71	4999.72	4999.73	4999.74	4999.75	4999.76	4999.77	4999.78	4999.79
4999.80	4999.81	4999.82	4999.83	4999.84	4999.85	4999.86	4999.87	4999.88	4999.89
4999.90	4999.91	4999.92	4999.93	4999.94	4999.95	4999.96	4999.97	4999.98	4999.99

Merge sorting of list (size=1000000) took 2.8274049758911133 sec

Before quickSort of L :

1716.75	-3404.87	-2562.36	2449.00	1839.78	-1379.78	2308.77	3337.70	-1074.36	1315.05
1873.79	-4905.41	1810.49	3252.46	4401.89	-172.90	-2632.81	-787.86	2997.89	1275.12
2786.31	-2858.59	2067.16	-888.97	1222.17	-4024.24	-2685.82	-894.66	-461.35	2049.85
.									
-802.01	-406.65	4998.22	14.46	1910.05	4350.52	-3818.31	1952.28	4456.15	4891.14
-1176.15	2486.11	3496.63	-2476.54	-1122.46	1253.70	4189.26	-4080.54	3985.78	1576.65
413.69	4777.83	3927.37	4452.39	3899.16	4945.32	-2963.02	1547.85	-3586.42	2931.75

After quickSort of L :

-5000.00	-4999.99	-4999.98	-4999.97	-4999.96	-4999.95	-4999.94	-4999.93	-4999.92	-4999.91
-4999.90	-4999.89	-4999.88	-4999.87	-4999.86	-4999.85	-4999.84	-4999.83	-4999.82	-4999.81
-4999.80	-4999.79	-4999.78	-4999.77	-4999.76	-4999.75	-4999.74	-4999.73	-4999.72	-4999.71
.									
4999.70	4999.71	4999.72	4999.73	4999.74	4999.75	4999.76	4999.77	4999.78	4999.79
4999.80	4999.81	4999.82	4999.83	4999.84	4999.85	4999.86	4999.87	4999.88	4999.89
4999.90	4999.91	4999.92	4999.93	4999.94	4999.95	4999.96	4999.97	4999.98	4999.99

Quick sorting of list (size=1000000) took 2.2479867935180664 sec



Generating random list of size (5000000) ...

Before mergeSort of L :

9488.59	-1191.63	15434.71	-4641.09	22833.81	-22371.68	510.14	-13697.12	12739.97	-15613.89
-1471.36	-1537.03	5182.48	-16166.02	2805.90	10231.45	-20346.92	-15599.43	-4641.98	477.12
-22421.08	-2047.88	20458.26	15866.49	2629.45	-13450.64	-15776.13	-9287.63	7336.32	15540.94
.									
17613.88	-22311.97	-22736.41	-18611.62	-15564.77	-1984.54	6279.26	-11552.99	-14484.76	-20314.71
13129.41	1536.20	-14051.02	9978.94	2912.64	21935.68	19664.60	7813.84	-18529.35	-11839.33
-5692.25	-18263.33	5220.10	-3125.05	-22054.54	-5734.21	-16834.25	13428.09	-7221.05	-13055.20

After mergeSort of L :

-25000.00	-24999.99	-24999.98	-24999.97	-24999.96	-24999.95	-24999.94	-24999.93	-24999.92	-24999.91
-24999.90	-24999.89	-24999.88	-24999.87	-24999.86	-24999.85	-24999.84	-24999.83	-24999.82	-24999.81
-24999.80	-24999.79	-24999.78	-24999.77	-24999.76	-24999.75	-24999.74	-24999.73	-24999.72	-24999.71
.									
24999.70	24999.71	24999.72	24999.73	24999.74	24999.75	24999.76	24999.77	24999.78	24999.79
24999.80	24999.81	24999.82	24999.83	24999.84	24999.85	24999.86	24999.87	24999.88	24999.89
24999.90	24999.91	24999.92	24999.93	24999.94	24999.95	24999.96	24999.97	24999.98	24999.99

Merge sorting of list (size=5000000) took 17.217018365859985 sec

Before quickSort of L :

-19296.14	-2705.46	-19849.81	15740.98	23554.56	-4445.18	7652.22	-18318.17	22880.11	19623.05
16269.45	-4795.56	-7542.58	-16993.18	-4623.19	-7222.81	-2931.38	5902.79	7162.33	-19942.99
-23401.82	18563.99	-17804.10	-16998.04	-20609.45	23275.99	18759.27	-19941.15	18316.58	20719.81
.									
-3083.55	-826.70	17773.76	3941.13	-20930.39	15302.18	-1427.70	-17758.04	-12512.06	10131.54
13426.84	-22497.03	4682.38	3990.35	12244.00	21846.01	8895.56	-22506.65	-1278.84	24457.03
7710.42	-10578.06	10857.06	5805.69	-2623.80	15221.23	-15987.49	-12613.76	22825.26	21657.52

After quickSort of L :

-25000.00	-24999.99	-24999.98	-24999.97	-24999.96	-24999.95	-24999.94	-24999.93	-24999.92	-24999.91
-24999.90	-24999.89	-24999.88	-24999.87	-24999.86	-24999.85	-24999.84	-24999.83	-24999.82	-24999.81
-24999.80	-24999.79	-24999.78	-24999.77	-24999.76	-24999.75	-24999.74	-24999.73	-24999.72	-24999.71
.									
24999.70	24999.71	24999.72	24999.73	24999.74	24999.75	24999.76	24999.77	24999.78	24999.79
24999.80	24999.81	24999.82	24999.83	24999.84	24999.85	24999.86	24999.87	24999.88	24999.89
24999.90	24999.91	24999.92	24999.93	24999.94	24999.95	24999.96	24999.97	24999.98	24999.99

Quick sorting of list (size=5000000) took 14.727692127227783 sec

