Assignment #2, Object Oriented Programming, Spring Semester, 2024

Due date: 2024.04.19

1. Write a program that dynamically creates an array of random size (between 5 and 20), fills each of its elements with random integers ranging from 0 to 100, and then prints all the values in the array, as well as the maximum and minimum values and the addresses of those maximum and minimum values. If there are two or more maximum or minimum values, print the address of the value with the lowest array index.

```
Ex)
Size of the array : 6
Random numbers : 86 87 82 62 46 14

Maximum value: 87          , Address: 000002922BA026D4
Minimum value: 14          , Address: 000002922BA026E4
```

2. Write a program that generates 10 random integers(0 to 100) and sorts the values in ascending order using **quick sort** and **merge sort**. Then implement a binary search algorithm to determine if the given integer exists (if not, insert it in the sorted position). Include in the report an analysis of sorting algorithms based on their time complexity.

```
Ex1)
Random values :   74 58 39 84 42 24 49 50 74 52
Select sorting method (1: Quick Sort, 2: Merge Sort) : 1
Sorted numbers (Quick Sort): 24 39 42 49 50 52 58 74 74 84
Enter a value to search : 50
Searched number index : 4
```

```
Ex2)
Random values :   74 58 39 84 42 24 49 50 74 52
Select sorting method (1: Quick Sort, 2: Merge Sort) : 1
Sorted numbers (Quick Sort): 24 39 42 49 50 52 58 74 74 84
Enter a value to search : 51
Updated numbers: 24 39 42 49 50 51 52 58 74 74 84
```

Assignment #2, Object Oriented Programming, Spring Semester, 2024

Due date: 2024.04.19

3. Write a program that prints a 10x10 matrix filled with random integers(0 to 100), stores the matrix in a variable of type int**, sorts and reprints the matrix by the **descending order** of the rows, and sorts and reprints the matrix by the **ascending order** of the sum of the rows. When sorting by the sum of the rows, you should replace the address pointed to by the pointer, not replace the value directly. Print the values with tab escape sequences between numbers (Please refer to the example below)

```
Original Matrix:
55      98      26      77      68      0       59      63      96      14
14      46      37      45      49      94      81      61      14      50
11      24      50      3       97      1       5       72      98      85
80      53      52      26      2       37      88      51      10      96
25      11      15      97      2       53      69      88      72      55
15      99      32      9       34      2       4       5       89      57
36      21      82      1       11      53      32      9       47      41
44      47      65      76      49      25      28      94      23      25
77      71      91      81      95      34      61      19      84      45
2       84      35      97      89      53      32      5       80      22

Sort by Row (Descending Order):
98      96      77      68      63      59      55      26      14      0       | Sum: 556
94      81      61      50      49      46      45      37      14      14      | Sum: 491
98      97      85      72      50      24      11      5       3       1       | Sum: 446
96      88      80      53      52      51      37      26      10      2       | Sum: 495
97      88      72      69      55      53      25      15      11      2       | Sum: 487
99      89      57      34      32      15      9       5       4       2       | Sum: 346
82      53      47      41      36      32      21      11      9       1       | Sum: 333
94      76      65      49      47      44      28      25      25      23      | Sum: 476
95      91      84      81      77      71      61      45      34      19      | Sum: 658
97      89      84      80      53      35      32      22      5       2       | Sum: 499

Sort by Sum (Ascending order) :
82      53      47      41      36      32      21      11      9       1       | Sum: 333
99      89      57      34      32      15      9       5       4       2       | Sum: 346
98      97      85      72      50      24      11      5       3       1       | Sum: 446
94      76      65      49      47      44      28      25      25      23      | Sum: 476
97      88      72      69      55      53      25      15      11      2       | Sum: 487
94      81      61      50      49      46      45      37      14      14      | Sum: 491
96      88      80      53      52      51      37      26      10      2       | Sum: 495
97      89      84      80      53      35      32      22      5       2       | Sum: 499
98      96      77      68      63      59      55      26      14      0       | Sum: 556
95      91      84      81      77      71      61      45      34      19      | Sum: 658
```

4. Write a program to separate strings by the given delimiters. Delimiters can be words, including spaces. (The use of the string library is prohibited)
   - The input string can be up to 100 characters
   - The input delimiter can be up to 10 characters

> Ex)
> Enter the string : C:₩Users₩Desktop₩2024₩OOP₩Assignment
> Enter the delimiter : ₩
>
> Separated tokens :
> C:
> User
> Desktop
> 2024
> OOP
> Assignment

Assignment #2, Object Oriented Programming, Spring Semester, 2024

Due date: 2024.04.19

5. A Hadamard matrix is a special form of a binary matrix in which every row and column has a different bit pattern. Write a program to generate a Hadamard matrix that satisfies the following conditions

   Step 1 : Accept an input n from the user. ('n' represents a power of 2)

   Step 2 : Generate a $2^n \times 2^n$ Hadamard matrix

   Step 3 : Display the generated Hadamard Matrix.

   Hadamard matrix is generated in a recursive way. Solve the problem using the method of using a smaller matrix to generate a larger matrix. The example below generates a Hadamard matrix of size 8x8

$$H_{2^0} = [1],\ H_{2^1} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix},\ H_{2^2} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}, H_{2^n} = \begin{bmatrix} H_{2^{n-1}} & H_{2^{n-1}} \\ H_{2^{n-1}} & -H_{2^{n-1}} \end{bmatrix}$$

```
Enter the value of n for Hadamard matrix (2^n x 2^n): 3

Hadamard Matrix of size 8x8:
1       1       1       1       1       1       1       1
1       -1      1       -1      1       -1      1       -1
1       1       -1      -1      1       1       -1      -1
1       -1      -1      1       1       -1      -1      1
1       1       1       1       -1      -1      -1      -1
1       -1      1       -1      -1      1       -1      1
1       1       -1      -1      -1      -1      1       1
1       -1      -1      1       -1      1       1       -1
```

Assignment #2, Object Oriented Programming, Spring Semester, 2024

Due date: 2024.04.19

6. Write a program that takes a text file as input and modifies it based on the implemented commands. The possible commands to implement are open, search, change, insert, delete, save, and exit.

Command	Format	Description
open	open [file path]	Open the file you want to edit. Exception handling should be performed in case the file does not exist.
search	search [word]	Print the starting column and row of all words found. Expressed as (row, column)
change	change [word1] [word2]	Change all of word1 to word2
insert	insert [row] [column] [word]	Add the words you want to add to the corresponding column and row positions
delete	delete [word]	Delete all 'word'
save	save [title]	Save the file with the title you entered
exit	exit	Exit the program

Input	Output
open ./testcase_6.txt search CHAPTER search glass change Alice Lucy insert 1 16 * delete CHAPTER save ./changedtext.txt exit	===‘CHAPTER’ search(1)=== (10, 28) --- ===‘glass’ search(3)=== (126, 6), (200, 26), (217, 32) --- ===change=== Alice -> Lucy --- ===insert=== * Inserted into (1, 16) --- ===delete=== Delete CHAPTER --- ===save=== Save the file as "./changedtext.txt" --- Exit the program

Assignment #2, Object Oriented Programming, Spring Semester, 2024

Due date: 2024.04.19

7. Write a calculator program that calculates a formula entered by the user. The calculator should support only integers and basic arithmetic operations (+, -, *, /). Write it so that it can handle parentheses. Utilize 'stack' to calculate the higher priority parts of the expression first. Implement and use the stack without using the stack library.
   - input numbers are between 0 and 255.
   - output number is between -2,147,843,648 and 2,147,483,647.
   - the number of operators is between 0 and 8.
   - the number of parentheses pair is between 0 and 8.

```
Enter the formula : 15 - ( 3 * 2 + 11 ) / 2
Result : 7
```

```
Enter the formula : 0
Result : 0
```

```
Enter the formula : 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9
Result : 45
```

```
Enter the formula : (((((((( 255 )))))))))
Result : 255
```

```
Enter the formula : (((((((( 1 ) - 2) + 3) - 4) + 5) - 6) + 7) - 8) + 9
Result : 5
```

Assignment #2, Object Oriented Programming, Spring Semester, 2024

Due date: 2024.04.19

8. Implement a 'Student' class with the members, as shown in the figure below, and create a program that performs actions based on the commands (insert, find, change, print, exit). The program should be able to store information of up to 10 students.
   - assignmentScore, examScore, attendance can range from 0 to 100
   - finalScore is calculated as 10% of attendance and 40% of assignmentScore and 50% of examScore
   - The name, studentID can be up to 100 characters
   - Case insensitive

Command	Format	Description
insert	insert [name] [studentID] [assignmentScore] [ExamScore] [attendance]	Add a new student object that stores the input information as its member variables through a constructor.
find	find [name]	If the name was found, print its information, otherwise print a 'not found' message.
change	change [name] [assignmentScore] [ExamScore] [attendance]	Change student [name]'s score information.
print	print	Print out the information of all stored students.
exit	exit	Exit the program.

```
class Student
{
private:
    char* name;
    char* studentID;
    double assignmentScore;
    double examScore;
    double attendance;
    double finalScore;

public:
    Student(char* name, char* id, double aScore, double eScore, double att);
    ~Student();
    bool isNameCorrect(char* name);
    void changeScores(double aScore, double eScore, double att);
    void print();
};
```

Input	Output
insert Kim 2024000001 80 80 100 insert Park 2024000002 50 75 80 insert Choi 2024000003 90 90 90 print find Lee find Kim change Kim 90 90 100 print exit	=====print===== Name : Kim Student ID : 2024000001 Final Score : 82 --------------- Name : Park Student ID : 2024000002 Final Score : 65.5 --------------- Name : Choi Student ID : 2024000003 Final Score : 90 --------------- =====find===== not found --------------- =====find===== Name : Kim Student ID : 2024000001 Final Score : 91 --------------- =====print===== Name : Kim Student ID : 2024000001 Final Score : 91 --------------- Name : Park Student ID : 2024000002 Final Score : 65.5 --------------- Name : Choi Student ID : 2024000003 Final Score : 90 --------------- Exit the program

Assignment #2, Object Oriented Programming, Spring Semester, 2024

Due date: 2024.04.19

9. Implement a program that stores and prints student information. To store student information, implement the Student Class below, and implement the School Class, which aims to manage multiple students' information. The Student Class should not be accessible outside of the School Class. Each class can only have the member variables shown in the figure below, and member functions can be implemented freely. The program should support six commands: new_student, sort_by_score, print_all, print_A_grade, print_B_grade, and exit.

   - The name, studentID can be up to 100 characters

```
class Studnet
{
private:
    char* name;
    char* studentID;
    double Score;
};
class School
{
private:
    class Student* student_list;
    int size = 0;
};
```

Command	Format	Description
new_student	new_student [name] [studentID] [Score]	Store the information of a new student in the school class.
sort_by_score	sort_by_score	Sort the student objects by Score in descending order.
print_all	print_all	Print information of all students stored in the School class.
print_A_grade	print_A_grade	Print information about students whose scores are in the top 30 percent
print_B_grade	print_B_grade	Print information for students whose scores are in the top 50% but are not A's
exit	exit	Terminate program

Input	Output
new_student Olivia 2013000005 95 new_student Emma 2013020103 55 new_student Amelia 2014100001 71 new_student Ava 2011020306 64 new_student Sophia 2013000004 80 print_all sort_by_score print_all exit	=====print===== Name : Olivia Student ID : 2013000005 Score : 95 --- Name : Emma Student ID : 2013020103 Score : 55 --- Name : Amelia Student ID : 2014100001 Score : 71 --- Name : Ava Student ID : 2011020306 Score : 64 --- Name : Sophia Student ID : 2013000004 Score : 80 --- =====print===== Name : Olivia Student ID : 2013000005 Score : 95 --- Name : Sophia Student ID : 2013000004 Score : 80 --- Name : Amelia Student ID : 2014100001 Score : 71 --- Name : Ava Student ID : 2011020306 Score : 64 --- Name : Emma Student ID : 2013020103 Score : 55 --- Exit the program

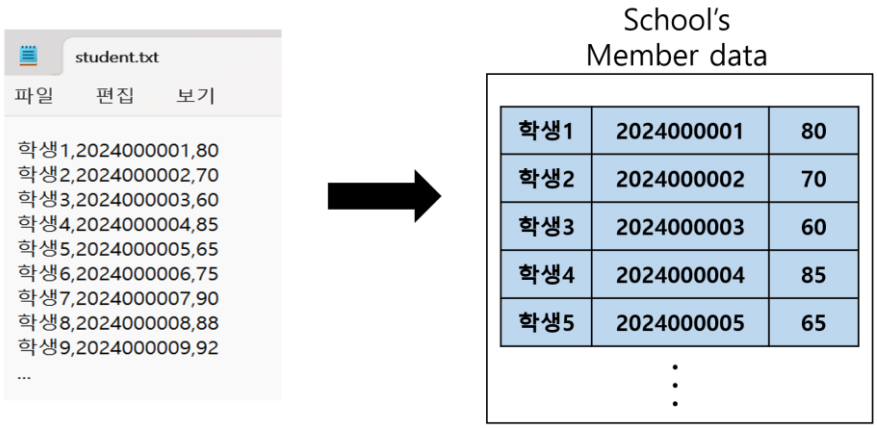
Input	Output
new_student Olivia 2013000005 95 new_student Emma 2013020103 55 new_student Amelia 2014100001 71 new_student Ava 2011020306 64 new_student Sophia 2013000004 80 print_A_grade print_B_grade exit	=====A grade===== Name : Olivia Student ID : 2013000005 Score : 95 --- =====B grade===== Name : Sophia Student ID : 2013000004 Score : 80 --- Exit the program

Assignment #2, Object Oriented Programming, Spring Semester, 2024

Due date: 2024.04.19

10. Write a program to read a 'student.txt' file and store information about students in a school class.
    - text files separate columns with a delimiter (,) and rows with newlines
    - Each row has 3 columns (name, studentID, score)
    - The name, studentID can be up to 100 characters

    change the 'new_student' command to load the student's information via cin at problem 9 to the 'load_student' command to load the student's information via a file.



Input	Output
load_student print_all sort_by_score print_all Exit	=====print===== Name : Olivia Student ID : 2013000005 Score : 95 --- Name : Emma Student ID : 2013020103 Score : 55 --- Name : Amelia Student ID : 2014100001 Score : 71 --- Name : Ava Student ID : 2011020306 Score : 64 --- Name : Sophia Student ID : 2013000004 Score : 80 --- =====print===== Name : Olivia Student ID : 2013000005 Score : 95 --- Name : Sophia Student ID : 2013000004 Score : 80 --- Name : Amelia Student ID : 2014100001 Score : 71 --- Name : Ava Student ID : 2011020306 Score : 64 --- Name : Emma Student ID : 2013020103 Score : 55 --- Exit the program

Assignment #2, Object Oriented Programming, Spring Semester, 2024

Due date: 2024.04.19

11. Write a program that uses a linked list to store integer values. The program has three commands that are insert, delete, and exit. You can see the details of each command in the table attached below. Please note that for the insert command, the linked list must insert the input value and sort the list in descending order simultaneously. And each time an insert command or a delete command is entered, print all the values in the list in order.

Command	Format	Description
insert	insert [number]	Add value to a Linked list and prints all the values in the list.
delete	delete [number]	Delete values from a Linked list and print all values in the list. (Delete all duplicate values.)
exit	exit	Exit the program.

Input	Output
insert 1	Linked list : 1
insert 2	Linked list : 2 -> 1
insert 3	Linked list : 3 -> 2 -> 1
insert 4	Linked list : 4 -> 3 -> 2 -> 1
insert 5	Linked list : 5 -> 4 -> 3 -> 2 -> 1
insert 6	Linked list : 6 -> 5 -> 4 -> 3 -> 2 -> 1
delete 4	Linked list : 6 -> 5 -> 3 -> 2 -> 1
delete 1	Linked list : 6 -> 5 -> 3 -> 2
exit	Exit the program

Assignment #2, Object Oriented Programming, Spring Semester, 2024

Due date: 2024.04.19

12. Write a program that stores English words (maximum size of a word is 15) in alphabetical order(case insensitive) like the English dictionary. To store the words, the program uses the 2D linked list as a data structure the first order of the list represents an alphabet and the second order of the list contains the words starting with the alphabet. The input of the program is given as input.dat and the output must be displayed to the standard output. The examples of input and output are given as follows:

Input format (Input.dat)

```
Apple
Mountain
Candle
Elephant
Notebook
Ocean
Tiger
Butterfly
Guitar
Laptop
River
Sunshine
Diamond
...
```

Output
A : Adult -> Advantage -> After -> Apple
B : Background -> Beak -> Beard -> Butterfly
C : Candle -> Cardboard
D : Desert -> Diamond -> Door
...