Algorithm Assignment#3

German language and literature
2016130927
Park Jun Yeong

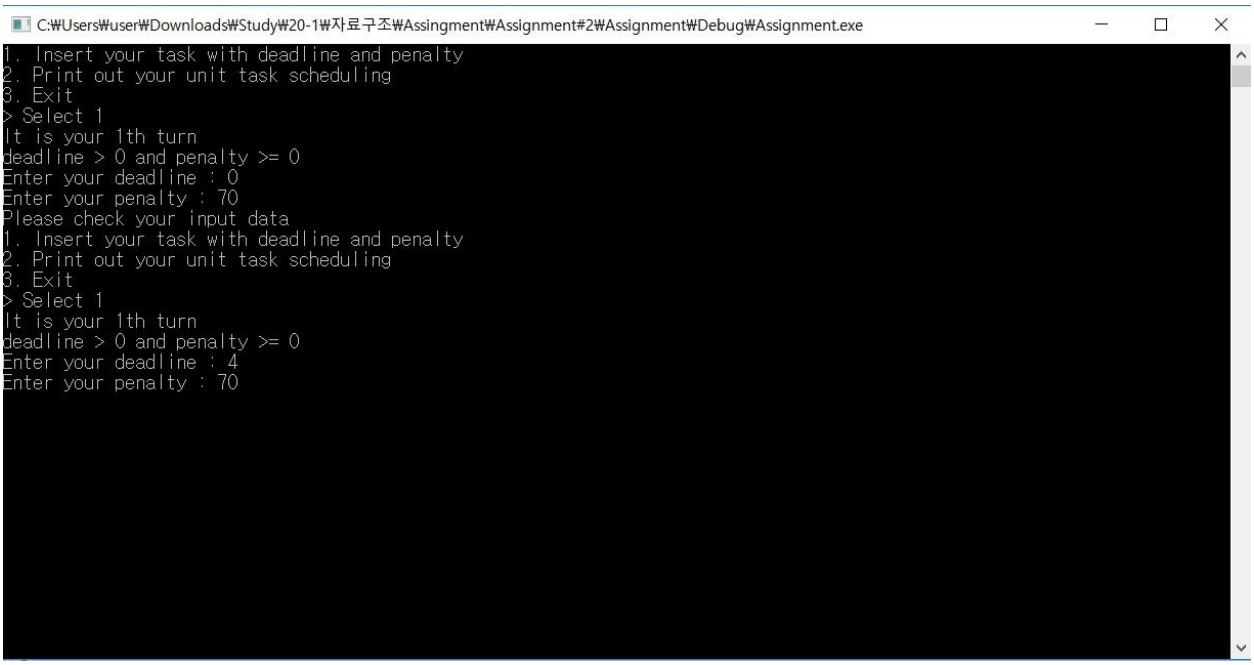
I. IDE: Visual Studio 2013

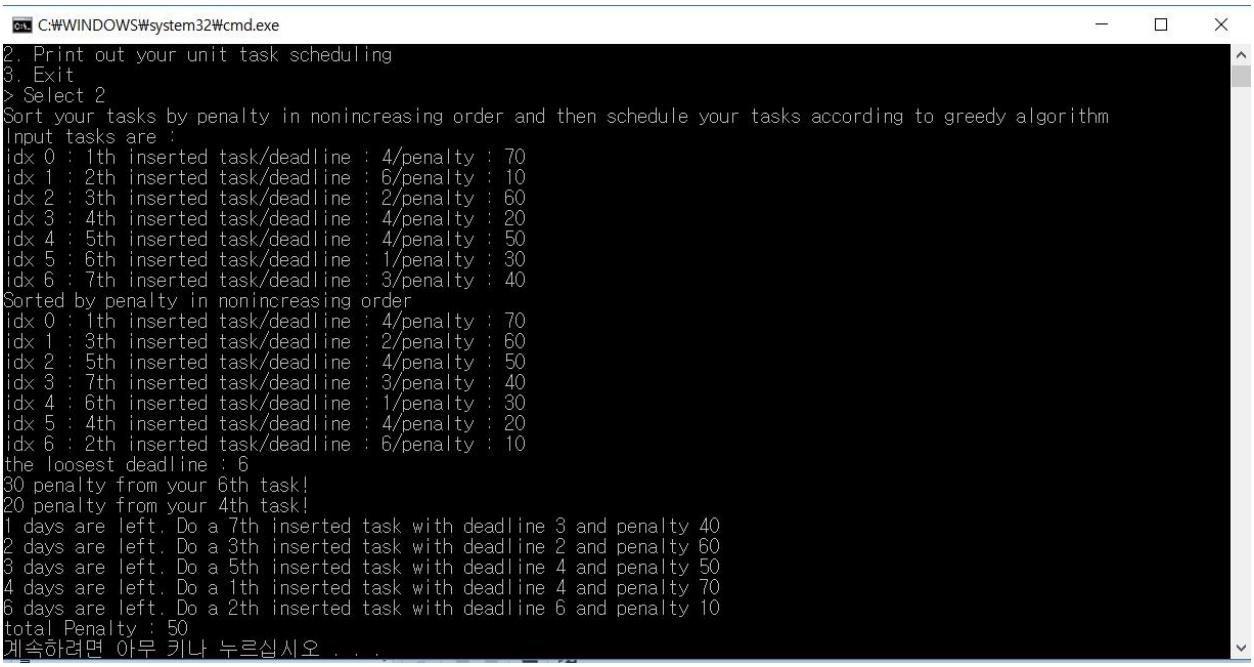
II. Test Data: this program requires you to enter the information. Below is an example.

1st task : deadline 4 penalty 70
2nd task : deadline 6 penalty 10
3rd task : deadline 2 penalty 60
4th task : deadline 4 penalty 20
5th task : deadline 4 penalty 50
6th task : deadline 1 penalty 30
7th task : deadline 3 penalty 40

These are entered and then after executing total penalty 50 is printed out. Below is the more details.

III. Screenshot of the execution result:





```
C:₩WINDOWS₩system32₩cmd.exe
2. Print out your unit task scheduling
3. Exit
> Select 2
Sort your tasks by penalty in nonincreasing order and then schedule your tasks according to greedy algorithm
Input tasks are :
idx 0 : 1th inserted task/deadline : 4/penalty : 70
idx 1 : 2th inserted task/deadline : 6/penalty : 10
idx 2 : 3th inserted task/deadline : 2/penalty : 60
idx 3 : 4th inserted task/deadline : 4/penalty : 20
idx 4 : 5th inserted task/deadline : 4/penalty : 50
idx 5 : 6th inserted task/deadline : 1/penalty : 30
idx 6 : 7th inserted task/deadline : 3/penalty : 40
Sorted by penalty in nonincreasing order
idx 0 : 1th inserted task/deadline : 4/penalty : 70
idx 1 : 3th inserted task/deadline : 2/penalty : 60
idx 2 : 5th inserted task/deadline : 4/penalty : 50
idx 3 : 7th inserted task/deadline : 3/penalty : 40
idx 4 : 6th inserted task/deadline : 1/penalty : 30
idx 5 : 4th inserted task/deadline : 4/penalty : 20
idx 6 : 2th inserted task/deadline : 6/penalty : 10
the loosest deadline : 6
30 penalty from your 6th task!
20 penalty from your 4th task!
1 days are left. Do a 7th inserted task with deadline 3 and penalty 40
2 days are left. Do a 3th inserted task with deadline 2 and penalty 60
3 days are left. Do a 5th inserted task with deadline 4 and penalty 50
4 days are left. Do a 1th inserted task with deadline 4 and penalty 70
6 days are left. Do a 2th inserted task with deadline 6 and penalty 10
total Penalty : 50
계속하려면 아무 키나 누르십시오 . . .
```

IV. Source C code with commentaries

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define MAX_SIZE 100

typedef struct
{
	int num;
	int deadline;
	int penalty;
}Unit; //a unit struct for your 'num'th inserted task with deadline and
penalty

typedef struct
{
	int mark;
	Unit * link;
}Set;
//a Set struct for checking whether a certain place (refering to deadline) is
visited or not and if not occupied yet, give it a task's address

int count = 0; //the number of tasks that you insert, which is set as 0
initially.
Unit sorted[MAX_SIZE];
/*another array for implementing mergesort of inserted tasks, which is a
global variable*/
```

```
void Merge(Unit unit[], int left, int mid, int right);
void MergeSort(Unit unit[], int left, int right);
int UTS(Unit unit[]);
void Print(Unit unit[]);
//functions for a unit task scheduling algorithm. more details are given
below.

int main(void)
{
	Unit unit[MAX_SIZE]; //a array of Unit struct composed of num,
deadline and penalty. num is automatically added and the others are entered.
	int sel; //which selects the operations from insertion to scheduling
to exit.
	int deadline, penalty;
	while (1)
	{
		printf("1. Insert your task with deadline and penalty\n");
		printf("2. Print out your unit task scheduling\n");
		printf("3. Exit\n");
		printf("> Select ");

		scanf_s("%d", &sel);
		switch (sel)
		{
		case 1: printf("It is your %dth turn\n", count+1);
			printf("deadline > 0 and penalty >= 0\n");
			unit[count].num = count+1;
			printf("Enter your deadline : ");
			scanf_s("%d", &deadline);
			printf("Enter your penalty : ");
			scanf_s("%d", &penalty);
			if (deadline <= 0 || penalty < 0)
			{
				printf("Please check your input data\n");
				continue;
			}
			unit[count].deadline = deadline;
			unit[count].penalty = penalty;
			printf("your %dth task is inserted\n", ++count);
			break;
			/*check whether deadline and penalty is positive integer.
(i thought penalty could be also zero)
			if not, continue and retry. if then, insert input into
idx of count. count is incremented by 1 after this operation.
			in other words, the number of tasks you have inserted is
count.*/

		case 2: printf("Sort your tasks by penalty in nonincreasing
```

```
order and then schedule your tasks according to greedy algorithm\n");
                    printf("Input tasks are :\n");
                    Print(unit);
                    MergeSort(unit, 0, count-1);
                    printf("Sorted by penalty in nonincreasing order\n");
                    Print(unit);
                    //before implemening greedy algorithm you have to sort
these tasks according to the size of penalty.
                    printf("total Penalty : %d\n", UTS(unit));
                    //UTS returns the total penalty, which prints out also
what kinds of tasks should be involved. if there's no input, just return -1.
                    system("pause");
                    break;

             case 3: printf("Exit this program\n");
                    exit(0);

             default: printf("Please selecte between 1~3\n");

             }
             system("cls");
      }
      return 0;
}

void Merge(Unit unit[], int left, int mid, int right)
{
      int i, j, k, l;
      i = left;
      j = mid + 1;
      k = left;

      while (i <= mid && j <= right)
      {
             if (unit[i].penalty >= unit[j].penalty)
                    sorted[k++] = unit[i++];
             else
                    sorted[k++] = unit[j++];
      }

      if (i>mid)
      {
             for (l = j; l <= right; l++)
                    sorted[k++] = unit[l];
      }

      else
      {
             for (l = i; l <= mid; l++)
```

```
                        sorted[k++] = unit[l];
        }
        for (l = left; l <= right; l++){
                unit[l] = sorted[l];
        }
}

void MergeSort(Unit unit[], int left, int right)
{
        int mid;

        if (left<right)
        {
                mid = (left + right) / 2;
                MergeSort(unit, left, mid);
                MergeSort(unit, mid + 1, right);
                Merge(unit, left, mid, right);
        }
}//your tasks are sorted by penalty in nonincresing order, implemented by
MergeSort and Merge function. Tc is O(nlgn)

int UTS(Unit unit[])
{
        if (count == 0) { printf("there is no data\n"); return -1; }

        Set *schedule;
        int total = 0;
        int i, j;
        int LD = unit[0].deadline;
        for (int i = 1; i < count; i++)
        {
                LD = (LD > unit[i].deadline) ? LD : unit[i].deadline;
        }
        /*which refers to the loosest deadline among your tasks. Since it is
required to have a enough place refering to deadline,
        from 1~ the loosest deadline so you need to find LD.*/
        printf("the loosest deadline : %d\n", LD);
        schedule = (Set *)malloc((LD+1)*sizeof(Set));
        /*allocate array of Set, Schedule dynamically. each Set is composed of
integer mark and Unit pointer link.
        Since we are going to check schedule's index as the appropriate
deadline of your task, idx '1~LD' could be used.
        in allocating size of (LD+1) is required to satisfy this condition. so
just ignore schedule[0].*/
        for (i = 1; i <= LD; i++)
                schedule[i].mark = 0;
        //marking unvisited places from schedule[1] to schedule[LD] with 0
initially.
        for (i = 0; i < count; i++)
```

```
		//from unit[0] to unit[count-1] (from the biggest penalty to the
smallest penalty)
	{
			for (j = unit[i].deadline; j > 0; j--)
				//since 'deadline(a certain number)' days are left
to submit, checking boundary is restricted from this 'deadline' day to 1.
			{
				if (schedule[j].mark == 0)
				{
					schedule[j].mark = 1;
					schedule[j].link = &unit[i];
					break;
					/*find unvisited place from its original
deadline to 1, which is decremented by 1.
					if unvisited, mark, give its address and
escape. if not (every place is full), j becomes 0*/
				}
			}
			if (j == 0)
			{
				printf("%d penalty from your %dth task!\n",
unit[i].penalty, unit[i].num);
				total += unit[i].penalty;
				//if every place is full, then you have to skip
that task. its penalty is added in total, checking all of possible penalties.
			}
	}

	for (i = 1; i <= LD; i++)
	{
		if (schedule[i].mark == 1)
			printf("%d days are left. Do a %dth inserted task with
deadline %d and penalty %d\n",
			i, schedule[i].link->num, schedule[i].link->deadline,
schedule[i].link->penalty);
	}//print out all of tasks that you should do.
	free(schedule);
	return total;
}

void Print(Unit unit[])
{
	if (count == 0)
	{
		printf("There is no data\n");
		return;
	}
	for (int i = 0; i < count; i++)
	{
```

```
            printf("idx %d : %dth inserted
task/deadline : %d/penalty : %d\n", i, unit[i].num, unit[i].deadline,
unit[i].penalty);
        }
        return;
}
```