Week7\_1a (Bubble sort)

The logic:

1. Every pair of adjacent numbers is examined by bubble sort as it cycles through the array of numbers.
2. The higher number will then be placed on the right, towards the end of the array, and the lower number will be placed on the left, towards the beginning.
3. And since we are sorting in ascending order, it will swap the numbers if ar[i]>ar[i+1].
4. At the conclusion of the first iteration, the array's largest number would be in the final position.
5. This is done repeatedly, and bubble sort will keep looping through the array until there are no swaps, leaving a sorted array.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 6 | 3 | 4 | 2 | 5 | 1 |

Unsorted array

**Example for the first pass (every pass will have the same logic)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 6 | 3 | 4 | 2 | 5 | 1 |

6>3, so 6 and 3 will swap position.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3 | 6 | 4 | 2 | 5 | 1 |

6>4, so 6 and 4 will swap position.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3 | 4 | 6 | 2 | 5 | 1 |

6>2, so 6 and 2 will swap position.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3 | 4 | 2 | 6 | 5 | 1 |

6>5, so 6 and 5 will swap position.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3 | 4 | 2 | 5 | 6 | 1 |

6>1, so 6 and 1 will swap position.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3 | 4 | 2 | 5 | 1 | 6 |

Result: 6 has moved to the right place.

**The diagram for the whole process:**

First pass: 6 is the largest, so 6 will be moved the last position.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3 | 4 | 2 | 5 | 1 | 6 |

Second pass: 5 is the largest, so 5 will be moved the last position before 6

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3 | 4 | 2 | 1 | 5 | 6 |

Third pass: 4 is the largest, so 4 will be moved the last position before 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3 | 2 | 1 | 4 | 5 | 6 |

Fourth pass: 3 is the largest, so 3 will be moved the last position before 4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2 | 1 | 3 | 4 | 5 | 6 |

Fifth pass: 2 is the largest, so 2 will be moved the last position before 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |

Sixth pass: no swapping needed

Week7\_1b (Selection sort)

The logic:

The element at the index where it should be is switched with the list's minimum (or maximum) element.

Repeating the steps results in a swap of the (n) minimum (or maximum) element with the element at the (n-1) index of the list.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 6 | 3 | 4 | 2 | 5 | 1 |

Unsorted array

Pass 1: which the minimum element 1 is selected and swapped with the element 6, at the lowest

index 0.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 2 | 5 | 6 |

Pass 2: only the sub-list is considered, excluding the element 1. So element 2, is swapped with element 3, in the 2nd lowest index position

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 4 | 3 | 5 | 6 |

Pass 3:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |

….

Last pass:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |

(This process is repeated until the sub-list is reduced to a single element in the highest index (i.e., its proper position)

Week7\_1c (Insertion sort)

The logic:

1. Start with the second element of the array.
2. Compare the second element with the first element. If the second element is smaller, swap them.
3. Move to the third element and compare it with the second element. If the third element is smaller, swap it with the second element. Now, compare the second element with the first element. If the second element is smaller, swap them.
4. Continue this process for all the remaining elements of the array, comparing each element with the elements before it and swapping them if necessary.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 6 | 3 | 4 | 2 | 5 | 1 |

Unsorted array

Pass 1:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3 | 6 | 4 | 2 | 5 | 1 |

Pass 2:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3 | 4 | 6 | 2 | 5 | 1 |

Pass 3:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2 | 3 | 4 | 6 | 5 | 1 |

Pass 4:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2 | 3 | 4 | 5 | 6 | 1 |

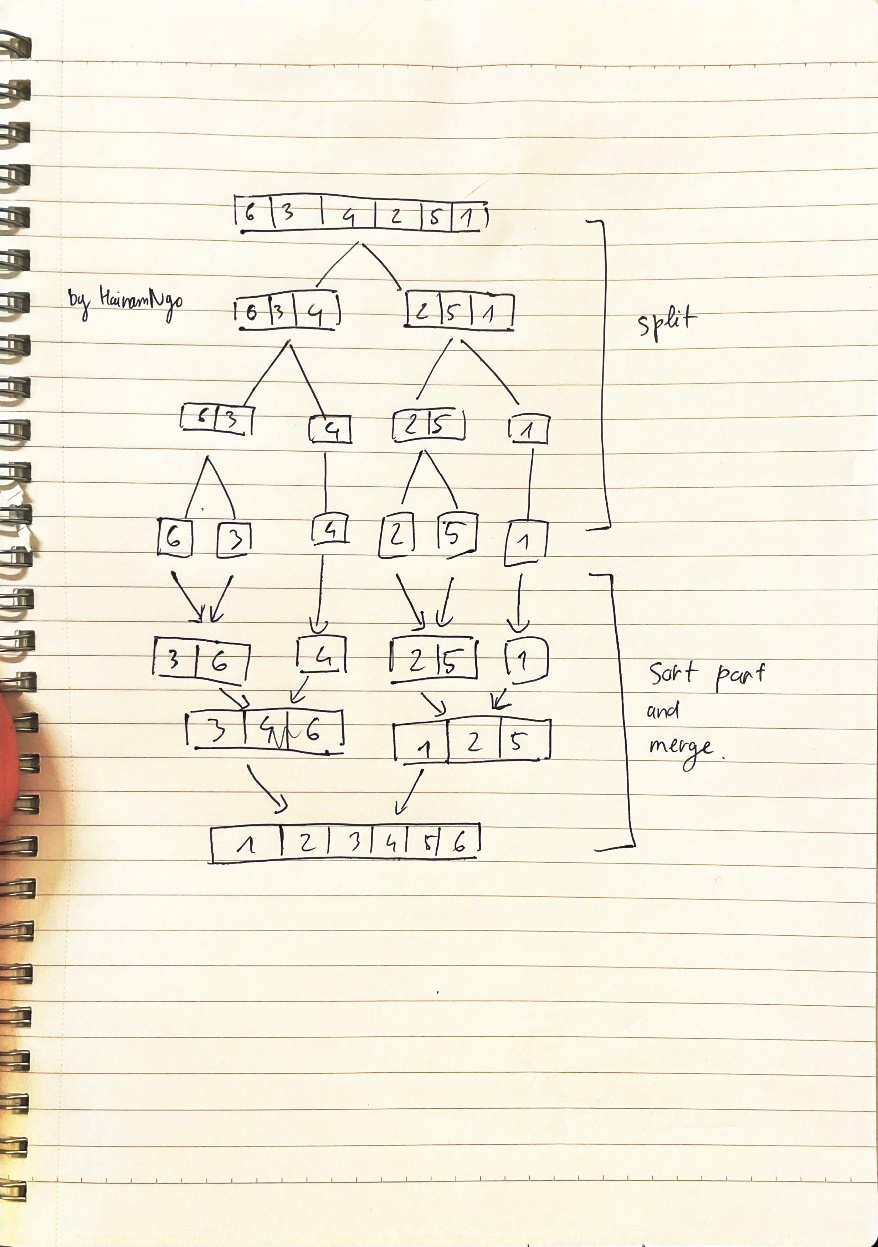
Pass 5:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |

Week7\_1d

The logic:

1. Divide the unsorted array into n subarrays, each containing one element (this is the base case).
2. Merge the subarrays in pairs to produce new sorted subarrays of twice the size.
3. Continue to merge subarrays in pairs until there is only one sorted array remaining.



Week7\_1e

The logic:

1. Choose a pivot element from the array.
2. Partition the array into two sub-arrays, one containing elements less than the pivot and the other containing elements greater than the pivot.
3. Recursively apply the above step to each of the sub-arrays until the entire array is sorted.

**Example:**

Choose 4 as the pivot element

Partition the array: [3, 2, 1] [4] [6, 5]

Recursively sort the sub-arrays: [1, 2, 3] [4] [5, 6]

Concatenate the sorted sub-arrays: [1, 2, 3, 4, 5, 6]

So the sorted array using Quick Sort is [1, 2, 3, 4, 5, 6].

W7\_2a.c

/\*

Unit Code: COS10007

Unit Name: Developing Techinical Software

Student ID: 103488515

Name: Hai Nam Ngo

Date Created: 04/20/2023

Date Modified: 04/20/2023

Problem: Week 7 Question 2a

Problem Description: create an array of size 5 students. Then write a

complete C program to sort the students array based on the students rank using bubble sort.

\*/

#include <stdio.h>

#include <stdlib.h>

struct student{

char name[10];

int rank;

};

#define SIZE 5

void BubbleSort(struct student a[], int array\_size)

{

int i, j;

struct student temp;

for (i = 0; i < (array\_size - 1); ++i)

{

for (j = 0; j < array\_size - 1 - i; ++j)

{

if (a[j].rank > a[j+1].rank)

{

temp = a[j+1];

a[j+1] = a[j];

a[j] = temp;

}

}

}

}

int main()

{

struct student a[SIZE];

for (int i=0;i<SIZE;i++)

{

printf("Enter student's name \n");

scanf("%s",a[i].name);

printf("Enter student's rank \n");

scanf("%d", &a[i].rank);

}

printf("The unsorted array is \n");

for(int i=0;i<SIZE;i++)

{

printf("Student's name: %s \t Rank %d \n",a[i].name,a[i].rank);

}

BubbleSort(a,SIZE);

printf("The sorted array is \n");

for(int i=0;i<SIZE;i++)

{

printf("Student's name: %s \t Rank %d \n",a[i].name,a[i].rank);

}

Graphical user interface, text

Description automatically generated return 0;

}

OUTPUT FOR W7\_2a.c

W7\_2b.c

/\*

Unit Code: COS10007

Unit Name: Developing Techinical Software

Student ID: 103488515

Name: Hai Nam Ngo

Date Created: 04/20/2023

Date Modified: 04/20/2023

Problem: Week 7 Question 2b

Problem Description: create an array of size 5 students. Then write a

complete C program to sort the students array based on the students rank using selection sort.

\*/

#include <stdio.h>

#include <stdlib.h>

struct student{

char name[10];

int rank;

};

#define SIZE 5

void SelectionSort(struct student a[], int array\_size)

{

int i;

for (i = 0; i < array\_size-1; ++i)

{

int j, min;

struct student temp;

min = i;

for (j = i+1; j < array\_size; ++j)

{

if (a[j].rank < a[min].rank)

min = j;

}

temp = a[i];

a[i] = a[min];

a[min] = temp;

}

}

int main()

{

struct student a[SIZE];

for (int i=0;i<SIZE;i++)

{

printf("Enter student's name \n");

scanf("%s",a[i].name);

printf("Enter student's rank \n");

scanf("%d", &a[i].rank);

}

printf("The unsorted array is \n");

for(int i=0;i<SIZE;i++)

{

printf("Student's name: %s and Rank %d",a[i].name,a[i].rank);

}

SelectionSort(a,SIZE);

printf("The sorted array is \n");

for(int i=0;i<SIZE;i++)

{

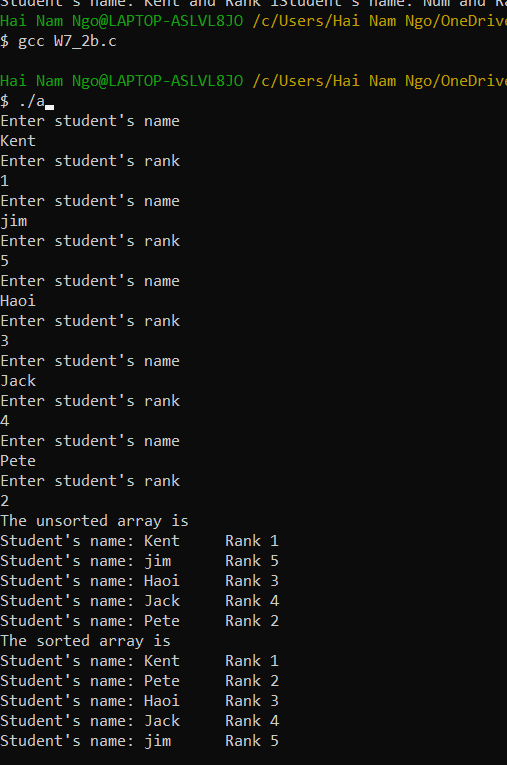
printf("Student's name: %s and Rank %d",a[i].name,a[i].rank);

}

return 0;

}

OUTPUT FOR W7\_2b.c



W7\_3a.c

/\*

Unit Code: COS10007

Unit Name: Developing Techinical Software

Student ID: 103488515

Name: Hai Nam Ngo

Date Created: 04/20/2023

Date Modified: 04/20/2023

Problem: Week 7 Question 3a

Problem Description: Using the veg structure, create an array of size 5. Then write a complete C

program to sort the array based on the item’s price by using Insertion sort.

\*/

#include <stdio.h>

#include <stdlib.h>

struct veg {

char item[10];

int price;

};

#define SIZE 5

void insertionSort(struct veg a[], int array\_size)

{

int i, j;

struct veg index;

for (i = 1; i < array\_size; ++i)

{

index = a[i];

for (j = i; j > 0 && a[j-1].price > index.price; j--)

a[j] = a[j-1];

a[j] = index;

}

}

int main()

{

struct veg a[SIZE];

for (int i=0;i<SIZE;i++)

{

printf("Enter the item: \n");

scanf("%s",a[i].item);

printf("Enter the price: \n");

scanf("%d", &a[i].price);

}

printf("The unsorted array is \n");

for(int i=0;i<SIZE;i++)

{

printf("Item: %s \t Price %d \n",a[i].item,a[i].price);

}

insertionSort(a,SIZE);

printf("The sorted array is \n");

for(int i=0;i<SIZE;i++)

{

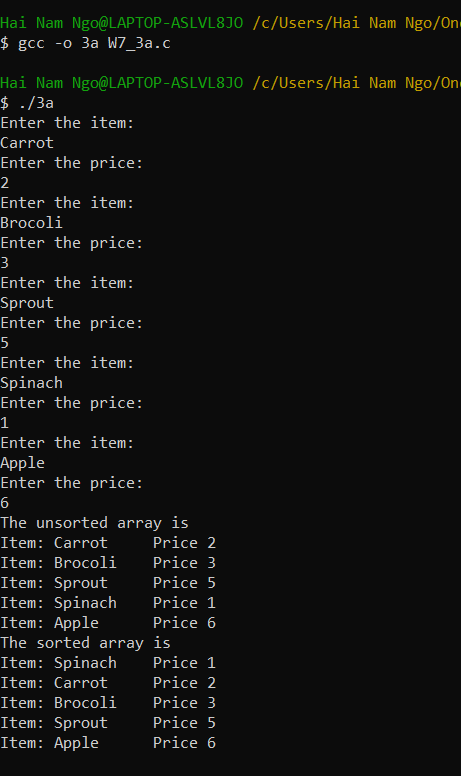
printf("Item: %s \t Price %d \n",a[i].item,a[i].price);

}

return 0;

}

OUTPUT FOR W7\_3a.c



W7\_3b.c

/\*

Unit Code: COS10007

Unit Name: Developing Techinical Software

Student ID: 103488515

Name: Hai Nam Ngo

Date Created: 04/20/2023

Date Modified: 04/20/2023

Problem: Week 7 Question 3b

Problem Description: Using the veg structure, create an array of size 5. Then write a complete C

program to sort the array based on the item’s price by using merge Sort

\*/

#include <stdio.h>

#include <stdlib.h>

struct veg {

char item[10];

int price;

};

#define SIZE 5

void Merge(struct veg A[],struct veg L[],int nL,struct veg R[],int nR) {

int i,j,k;

i = 0; j = 0; k =0;

while(i<nL && j< nR)

{

if(L[i].price < R[j].price)

{

A[k] = L[i];

i++;

}

else

{

A[k] = R[j];

j++;

}

k++;

}

while(i < nL)

{

A[k] = L[i];

k++;

i++;

}

while(j < nR)

{

A[k] = R[j];

k++;

j++;

}

}

void MergeSort(struct veg \*A,int n) {

int mid,i;

if(n < 2) return;

mid = n/2;

struct veg \*L = (struct veg\*) malloc(mid\*sizeof(struct veg));

struct veg \*R = (struct veg\*) malloc((n - mid)\*sizeof(struct veg));

for(i = 0;i<mid;i++)

L[i] = A[i];

for(i = mid;i<n;i++)

R[i-mid] = A[i];

MergeSort(L,mid);

MergeSort(R,n-mid);

Merge(A,L,mid,R,n-mid);

free(L);

free(R);

}

int main()

{

struct veg A[SIZE];

for (int i=0;i<SIZE;i++)

{

printf("Enter the item: \n");

scanf("%s",A[i].item);

printf("Enter the price: \n");

scanf("%d", &A[i].price);

}

printf("The unsorted array is \n");

for(int i=0;i<SIZE;i++)

{

printf("Item: %s \t\t Price %d \n",A[i].item,A[i].price);

}

MergeSort(A,SIZE);

printf("The sorted array is \n");

for(int i=0;i<SIZE;i++)

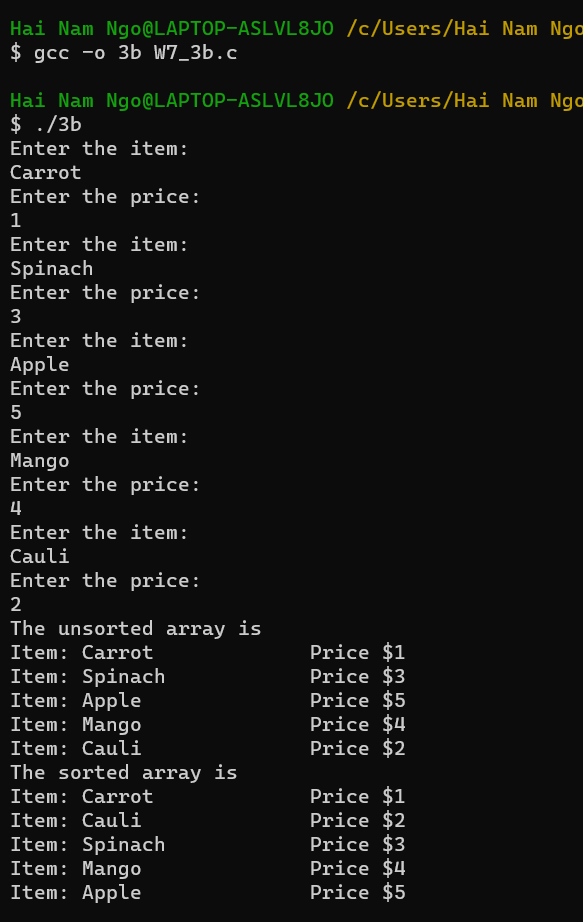
{

printf("Item: %s \t\t Price %d \n",A[i].item,A[i].price);

}

return 0;

}

OUTPUT FOR W7\_3b.c

W7\_3c.c

/\*

Unit Code: COS10007

Unit Name: Developing Techinical Software

Student ID: 103488515

Name: Hai Nam Ngo

Date Created: 04/20/2023

Date Modified: 04/20/2023

Problem: Week 7 Question 3c

Problem Description: Using the veg structure, create an array of size 5. Then write a complete C

program to sort the array based on the item’s price by using Quick sort.

\*/

#include <stdio.h>

#include <stdlib.h>

struct veg {

char item[10];

int price;

};

#define SIZE 5

void swap(struct veg\* a, struct veg\* b)

{

struct veg temp = \*a;

\*a = \*b;

\*b = temp;

}

int partition(struct veg array[], int start, int end)

{

struct veg pivot = array[end];

int pIndex=start;

for(int i=start; i < end; i++)

{

if(array[i].price <= pivot.price)

{

swap(&array[i],&array[pIndex]);

pIndex++;

}

}

swap(&array[pIndex],&array[end]);

return pIndex;

}

void Quicksort (struct veg array[], int start, int end)

{

if (start<end)

{

int pIndex=partition(array, start, end);

Quicksort(array, start, pIndex-1);

Quicksort(array, pIndex+1, end);

}

}

int main()

{

struct veg array[SIZE];

int start = 0, end = SIZE - 1;

for (int i=0;i<SIZE;i++)

{

printf("Enter the item: \n");

scanf("%s",array[i].item);

printf("Enter the price: \n");

scanf("%d", &array[i].price);

}

printf("The unsorted array is \n");

for(int i=0;i<SIZE;i++)

{

printf("Item: %s \t Price $%d \n",array[i].item,array[i].price);

}

Quicksort(array,start,end);

printf("The sorted array is \n");

for(int i=0;i<SIZE;i++)

{

printf("Item: %s \t Price $%d \n",array[i].item,array[i].price);

}

return 0;

}

OUTPUT FOR W7\_3c.c

