DASF004

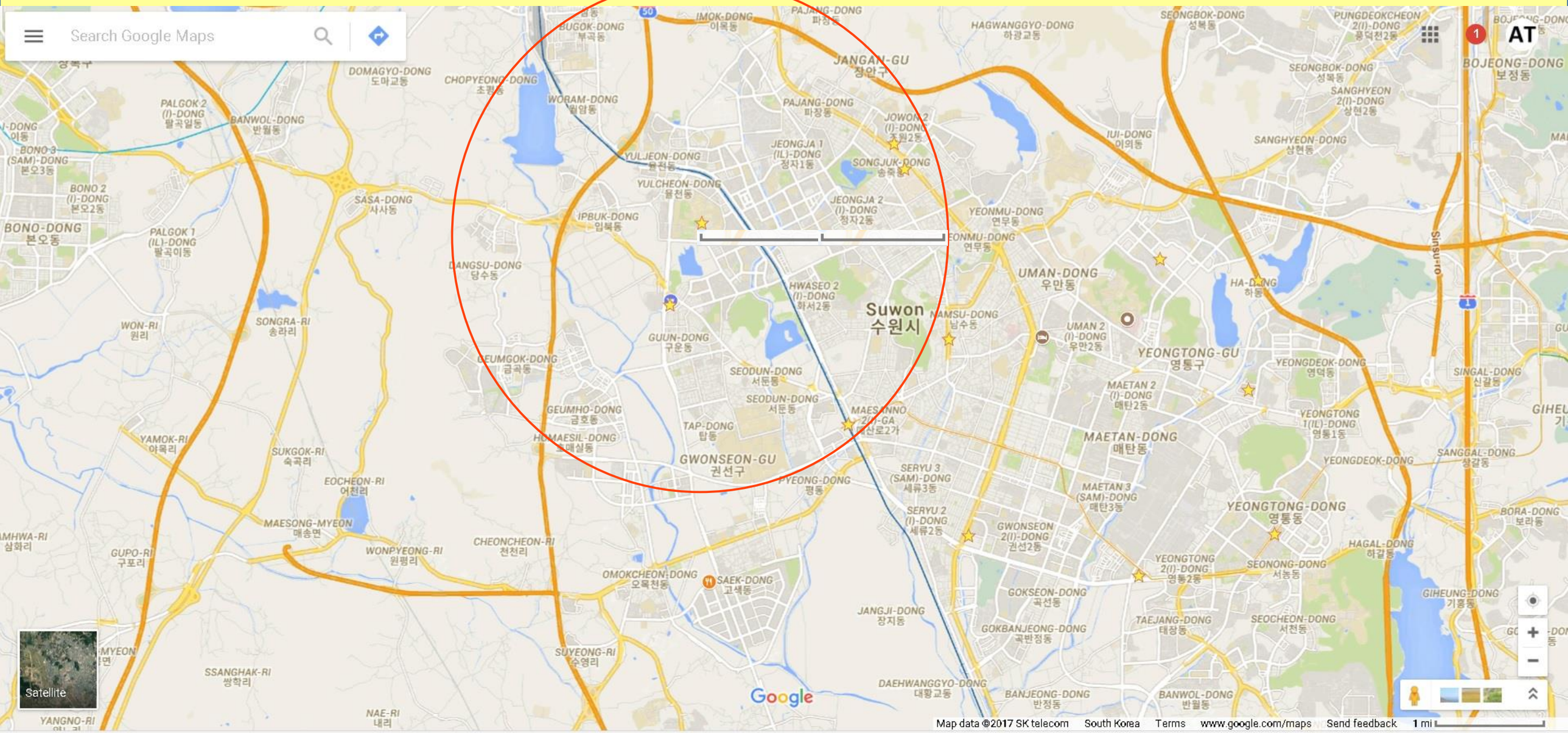
Basic and Practice in Programming

Lecture 9

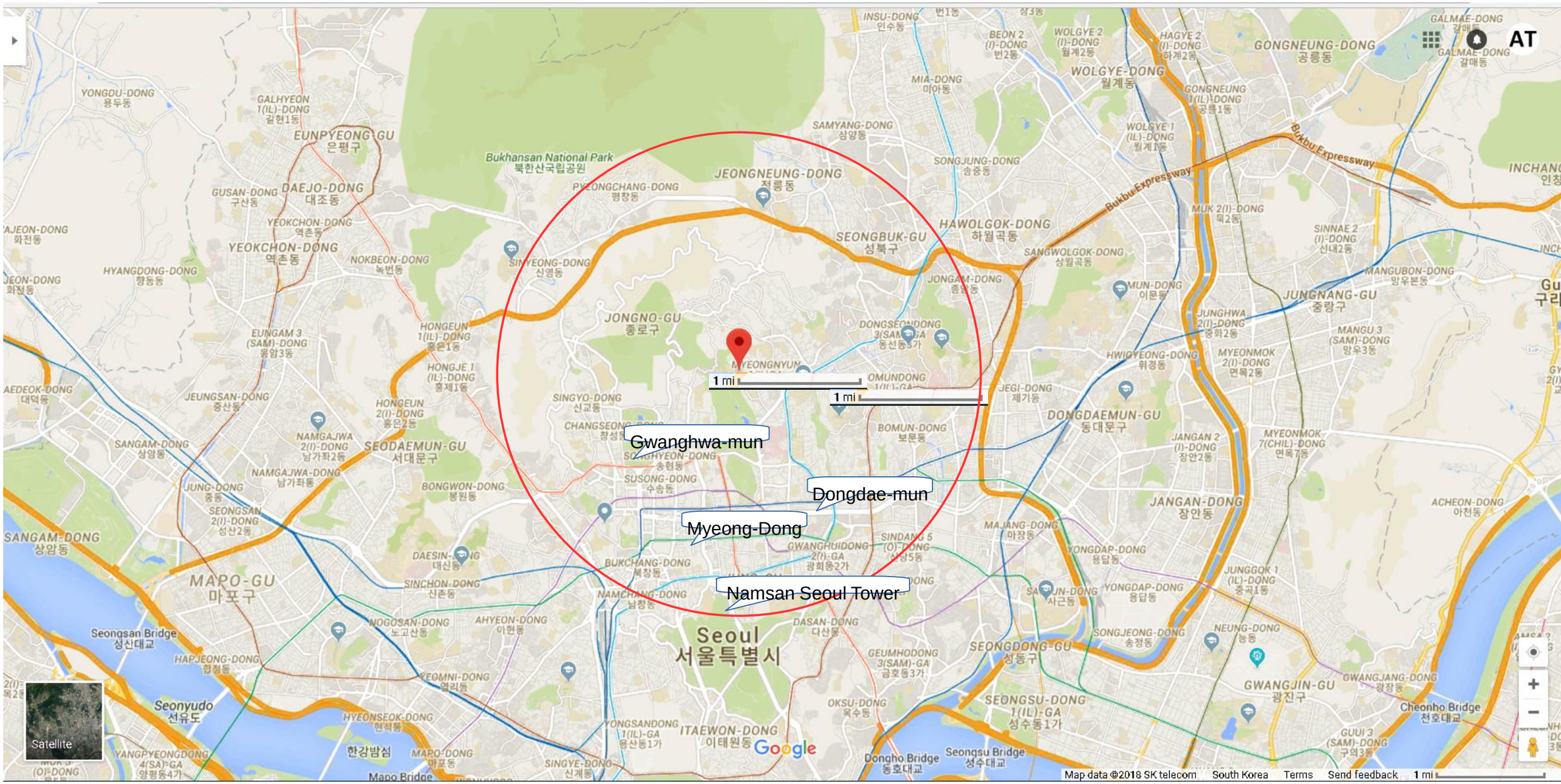
Pointer 2

Food for your MIND:
Intelligence Augmentation

How long is 2 miles?



How long is 2 miles?



Food for our MIND:
Intelligence Augmentation

While officials would not say where the shot took place, the statement noted the command "provides its expertise to Iraqi security forces to detect, identify and defeat Daesh activities from well behind the Iraqi security force front line in Mosul."

The new record was set using a McMillan TAC-50, a .50-caliber weapon and the largest shoulder-fired firearm in existence.

Ryan Cleckner, a former U.S. Army Ranger sniper who served two tours of duty in Afghanistan and wrote the authoritative “Long Range Shooting Handbook,” called the feat an “incredible” accomplishment, one that owes as much if not more to the spotter’s expertise as to the shooter's skill.

Related Image

The McMillan TAC-50 is a .50-caliber weapon, and the largest shoulder-fired firearm in existence (McMillan Firearms)

Related Image

Lebanese army snipers take their positions on the top of a building, in Tripoli in 2016. (Associated Press)

“The spotter would have had to successfully calculate five factors: distance, wind, atmospheric conditions and the speed of the earth’s rotation at their latitude,” Cleckner told Fox News.

“Because wind speed and direction would vary over the two miles the bullet traveled, the true challenge here was being able to calculate the actual wind speed and direction all the way to the target.”

Atmospheric conditions also would have posed a huge challenge for the spotter.

“To get the atmospheric conditions just right, the spotter would have had to understand the temperature, humidity and barometric pressure of the air the round had to travel through.

Cleckner said that while the ammunition that Canadian special forces use in the TAC-50 is “off-the-charts powerful,” with some

Related Image

Feb. 3, 2013: A U.S. Army sniper with Charlie Company, 36th Infantry Regiment, 1st Armored Division looks down the scope of his rifle during a mission near Command Outpost Pa'in Kalay in Maiwand District, Kandahar

Precision Guided Firearm (PGF)

https://www.youtube.com/watch?v=q0oGZ4TZr5k

https://youtu.be/Pmteh_NChOQ

Administration

Midterm Assignment

- Video is up
- Score will be available on iCampus VERY soon

Holiday on Wednesday

- University policy requires a makeup class in the case of Holiday
- Lab will be on Thursday (I may put the video up earlier, the deadline for lab exercise will be Thursday 23:59 pm).

Agenda

Pointer

- Pointer and Array
- Pointer: Applications

A Note on the Dereferencing operator *

* has a higher precedence than other arithmetic operators (+ - * / %)

The difference between *x+1 and *(x+1)

Pointer and Array

When you do an array declaration

```
int x[4] = {1,2,3,4};
```

- Memory for 4 integers is allocated
- A point variable x is created and pointing to the beginning of the memory location

Value	Address
x[0]: 1	6087928
x[1]: 2	6087932
x[2]: 3	6087936
x[3]: 4	6087940
x: 6087928	6087944

Pointer and Array

Value	Address
x[0]: 1	6087928
x[1]: 2	6087932
x[2]: 3	6087936
x[3]: 4	6087940
x: 6087928	6087944

The following 2 code segments are identical:

```
int x[4] = {1,2,3,4};
printf("%d\n",x[0]);
printf("%d\n",x[1]);
printf("%d\n",x[2]);
printf("%d\n",x[3]);
```

```
int x[4] = {1,2,3,4};
printf("%d\n",*x);
printf("%d\n",*(x+1));
printf("%d\n",*(x+2));
printf("%d\n",*(x+3));
```

You may treat the indexed form of an array as a short form for the pointer form

```
x[0] => *(x+0)
x[1] => *(x+1)
x[2] => *(x+2)
x[n] => *(x+n)
```

Three examples

1. Sorting an array
2. Searching a target in an array
3. Simulating a Deck of Playcard

Reminder

Good Programming Practice ……

Good Programming Style 3.3

For novice programmer, you can start writing a loop as a simple loop, then build your code based on the simple loop.

```
for(i=0; i<9; i++)
{ printf(“i: %d\n”, i);
}
```

Good Programming Style 3.4

When you got lost in coding the loop, print out the counter variable and other intermediate variables so that you know how many times the loop body has executed and the value of each variables.

```
while (counter <= 10)
{ scanf("Input: %f", Input);
  Sum = Sum + Input;
  printf("While loop %d times, Sum=%f\n", counter, Sum);
  count++;
}
```

Pointer example: Bubble sort

The Problem:

- You have a numerical array (int, float, double, etc)
- With unspecified number of items (from 0 to n items),
- i.e. you don't know how many items
- They are unsorted (not in any order)
  - e.g. {1,3,5,2,4,6}
- You want to create a general function to sort this array in ascending order
  - For example:

```
int x[6] = {1,3,5,2,4,6};
SortArray(x,6);
```

    - Turning {1,3,5,2,4,6} into {1,2,3,4,5,6}

- How do you do it?

Bubble sort: The Algorithm

1. Starting from the beginning, compare two adjacent items and arrange them in ascending order
2. Repeat this n-1 times

Bubble sort: The Algorithm

1. Starting from the beginning, compare two adjacent items and arrange them in ascending order
2. Repeat this n-1 times

5 items, repeat 4 passes

Number of comparison in each pass is 4

9 8 4 2 1

Bubble sort: The Algorithm

1. Starting from the beginning, compare two adjacent items and arrange them in ascending order
2. Repeat this n-1 times

Pass #1

```
9  8  4  2  1
^  ^
```

Compare => Swap!

Bubble sort: The Algorithm

1. Starting from the beginning, compare two adjacent items and arrange them in ascending order
2. Repeat this n-1 times

Pass #1

```
8  9  4  2  1
   ^  ^
```

Compare => Swap!

Bubble sort: The Algorithm

1. Starting from the beginning, compare two adjacent items and arrange them in ascending order
2. Repeat this n-1 times

Pass #1

```
8  4  9  2  1
      ^  ^
```

Compare => Swap!

Bubble sort: The Algorithm

1. Starting from the beginning, compare two adjacent items and arrange them in ascending order
2. Repeat this n-1 times

Pass #1

```
8  4  2  9  1
         ^  ^
```

Compare => Swap!

Bubble sort: The Algorithm

1. Starting from the beginning, compare two adjacent items and arrange them in ascending order
2. Repeat this n-1 times

Pass #1

8 4 2 1 9

End of Pass #1

Bubble sort: The Algorithm

1. Starting from the beginning, compare two adjacent items and arrange them in ascending order
2. Repeat this n-1 times

Pass #2

```
8  4  2  1  9
^  ^
```

Compare => Swap!

Bubble sort: The Algorithm

1. Starting from the beginning, compare two adjacent items and arrange them in ascending order
2. Repeat this n-1 times

Pass #2

```
4  8  2  1  9
   ^  ^
```

Compare => Swap!

Bubble sort: The Algorithm

1. Starting from the beginning, compare two adjacent items and arrange them in ascending order
2. Repeat this n-1 times

Pass #2

```
4  2  8  1  9
      ^  ^
```

Compare => Swap!

Bubble sort: The Algorithm

1. Starting from the beginning, compare two adjacent items and arrange them in ascending order
2. Repeat this n-1 times

Pass #2

```
4  2  1  8  9
      ^  ^
```

Compare => No Change

Bubble sort: The Algorithm

1. Starting from the beginning, compare two adjacent items and arrange them in ascending order
2. Repeat this n-1 times

Pass #2

```
4  2  1  8  9
      ^  ^
```

End of Pass #2

Bubble sort: The Algorithm

1. Starting from the beginning, compare two adjacent items and arrange them in ascending order
2. Repeat this n-1 times

Pass #3

```
4  2  1  8  9
^  ^
```

Compare => Swap!

Bubble sort: The Algorithm

1. Starting from the beginning, compare two adjacent items and arrange them in ascending order
2. Repeat this n-1 times

Pass #3

```
2  4  1  8  9
   ^  ^
```

Compare => Swap!

Bubble sort: The Algorithm

1. Starting from the beginning, compare two adjacent items and arrange them in ascending order
2. Repeat this n-1 times

Pass #3

```
2  1  4  8  9
      ^  ^
```

Compare => No Change

Bubble sort: The Algorithm

1. Starting from the beginning, compare two adjacent items and arrange them in ascending order
2. Repeat this n-1 times

Pass #3

```
2  1  4  8  9
      ^  ^
```

Compare => No Change

Bubble sort: The Algorithm

1. Starting from the beginning, compare two adjacent items and arrange them in ascending order
2. Repeat this n-1 times

Pass #3

```
2  1  4  8  9
      ^  ^
```

End of Pass #3

Bubble sort: The Algorithm

1. Starting from the beginning, compare two adjacent items and arrange them in ascending order
2. Repeat this n-1 times

Pass #4

```
2  1  4  8  9
^  ^
```

Compare => Swap!

Bubble sort: The Algorithm

1. Starting from the beginning, compare two adjacent items and arrange them in ascending order
2. Repeat this n-1 times

Pass #4

```
1  2  4  8  9
   ^  ^
```

Compare => No Change

Bubble sort: The Algorithm

1. Starting from the beginning, compare two adjacent items and arrange them in ascending order
2. Repeat this n-1 times

Pass #4

```
1  2  4  8  9
      ^  ^
```

Compare => No Change

Bubble sort: The Algorithm

1. Starting from the beginning, compare two adjacent items and arrange them in ascending order
2. Repeat this n-1 times

Pass #4

```
1  2  4  8  9
         ^  ^
```

Compare => No Change

Bubble sort: The Algorithm

1. Starting from the beginning, compare two adjacent items and arrange them in ascending order
2. Repeat this n-1 times

Pass #4

```
1  2  4  8  9
```

End of Pass #4

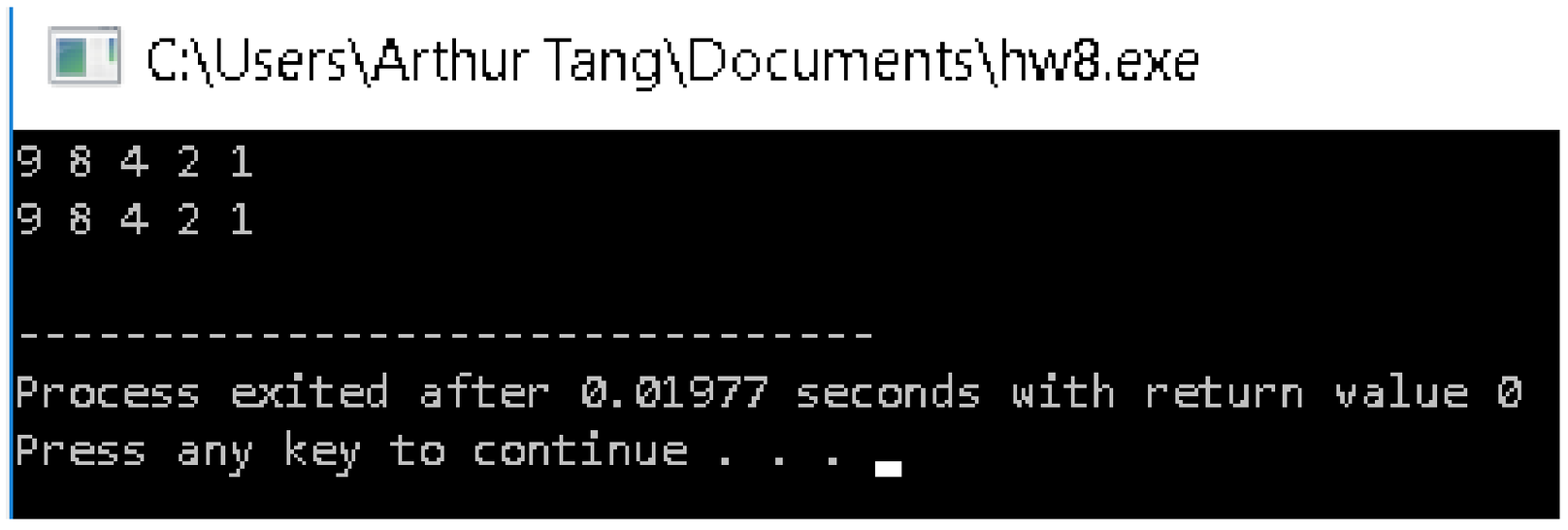
The Array is now sorted!!!

Bubble sort: How do you write this function???

```
#include <stdio.h>

void BubbleSort(int * x, int n)
{ // How do you write this function???
}

int main(void)
{ int test[5] = {9,8,4,2,1};
  BubbleSort(test,sizeof(test)/sizeof(int));
  printf("%d %d %d %d %d\n",test[0],test[1],test[2],test[3],test[4]);
  return 0;
}
```

Bubble sort: How do you write this function???

```
#include <stdio.h>

void BubbleSort(int * x, int n)
{ for(int i=0; i<n; i++)
    printf("%d ",x[i]);
  printf("\n");
  // Check if the array is passed to the function properly
}

int main(void)
{ int test[5] = {9,8,4,2,1};
  BubbleSort(test,sizeof(test)/sizeof(int));
  printf("%d %d %d %d %d\n",test[0],test[1],test[2],test[3],test[4]);
  return 0;
}
```



Bubble sort: How do you write this function???

```
#include <stdio.h>

void BubbleSort(int * x, int n)
{ for(int i=0; i<n-1; i++)
  {
  } // A loop looping n-1 times (Number of pass = n-1)
}
```

```
int main(void)
{ int test[5] = {9,8,4,2,1};
  BubbleSort(test,sizeof(test)/sizeof(int));
  printf("%d %d %d %d %d\n",test[0],test[1],test[2],test[3],test[4]);
  return 0;
}
```

Bubble sort: How do you write this function???

```
#include <stdio.h>

void BubbleSort(int * x, int n)
{ for(int i=0; i<n-1; i++)
  { for (int j=0; j<n-1; j++)
    {
    }  // Making n-1 comparison in each pass
  }   // A loop looping n-1 times (Number of pass = n-1)
}

int main(void)
{ int test[5] = {9,8,4,2,1};
  BubbleSort(test,sizeof(test)/sizeof(int));
  printf("%d %d %d %d %d\n",test[0],test[1],test[2],test[3],test[4]);
  return 0;
}
```

Bubble sort: How do you write this function???

```
#include <stdio.h>

void Swap(int * x, int * y)
{
}

void BubbleSort(int * x, int n)
{ for(int i=0; i<n-1; i++)
  { for (int j=0; j<n-1; j++)
    { if(x[j] > x[j+1])     // if adjacent items are not in order
        Swap(&x[j],&x[j+1]); // then swap them!
    }  // Making n-1 comparison in each pass
  }   // A loop looping n-1 times (Number of pass = n-1)
}

int main(void)
{ int test[5] = {9,8,4,2,1};
  BubbleSort(test,sizeof(test)/sizeof(int));
  printf("%d %d %d %d %d\n",test[0],test[1],test[2],test[3],test[4]);
  return 0;
}
```

Bubble sort: Completed!!!

```
#include <stdio.h>

void Swap(int * x, int * y)
{ int temp = *x;
  *x = *y;
  *y = temp;
}

void BubbleSort(int * x, int n)
{ for(int i=0; i<n-1; i++)
  { for (int j=0; j<n-1; j++)
    { if(x[j] > x[j+1])     // if adjacent items are not in order
        Swap(&x[j],&x[j+1]); // then swap them!
    }  // Making n-1 comparison in each pass
  }   // A loop looping n-1 times (Number of pass = n-1)
}

int main(void)
{ int test[5] = {9,8,4,2,1};
  BubbleSort(test,sizeof(test)/sizeof(int));
  printf("%d %d %d %d %d\n",test[0],test[1],test[2],test[3],test[4]);
  return 0;
}
```

Bubble sort: A few notes on this solution

- The argument array in BubbleSort is passing by reference; the modification remains.
- The argument array in Swap is also passing by reference; the modification also remains.
- Size of the array is passed as argument, making the function more generic for array of any size.
- The implementation also uses a modular approach, making the readability of the code better. It is also easier to debug.

How do you search for an item in an array?

Simplest solution: Exhaustion Search

- Search the array item one by one

From Bubble Sort ...

```
#include <stdio.h>

void Swap(int * x, int * y)
{ int temp = *x;
  *x = *y;
  *y = temp;
}

void BubbleSort(int * x, int n)
{ for(int i=0; i<n-1; i++)
    for (int j=0; j<n-1; j++)
      if(x[j] > x[j+1])      // if adjacent items are not in order
        Swap(&x[j],&x[j+1]); // then swap them!
}

int main(void)
{ int test[5] = {9,8,4,2,1};
  BubbleSort(test,sizeof(test)/sizeof(int));
  printf("%d %d %d %d %d\n",test[0],test[1],test[2],test[3],test[4]);
  return 0;
}
```

```c
#include <stdio.h>

void Swap(int * x, int * y)
{ int temp = *x;
  *x = *y;
  *y = temp;
}

void BubbleSort(int * x, int n)
{ for(int i=0; i<n-1; i++)
    for (int j=0; j<n-1; j++)
      if(x[j] > x[j+1])     // if adjacent items are not in order
        Swap(&x[j],&x[j+1]); // then swap them!
}

int * Search(int target, int * array, int size)
{ // How to implement the search function???
}

int main(void)
{ int test[5] = {9,8,4,2,1};
  BubbleSort(test,sizeof(test)/sizeof(int));
  printf("%d %d %d %d %d\n",test[0],test[1],test[2],test[3],test[4]);
  int target = 4, * targetPtr = NULL;
  targetPtr = Search(target,test,sizeof(test)/sizeof(int));
  return 0;
}
```

```c
#include <stdio.h>

void Swap(int * x, int * y)
{ int temp = *x;
  *x = *y;
  *y = temp;
}

void BubbleSort(int * x, int n)
{ for(int i=0; i<n-1; i++)
    for (int j=0; j<n-1; j++)
      if(x[j] > x[j+1])     // if adjacent items are not in order
        Swap(&x[j],&x[j+1]); // then swap them!
}
```

```c
int * Search(int target, int * array, int size)
{ for(int i=0;i<size;i++)  // A loop looping n times (n=size)
  {
  }
}

int main(void)
{ int test[5] = {9,8,4,2,1};
  BubbleSort(test,sizeof(test)/sizeof(int));
  printf("%d %d %d %d %d\n",test[0],test[1],test[2],test[3],test[4]);
  int target = 4, * targetPtr = NULL;
  targetPtr = Search(target,test,sizeof(test)/sizeof(int));
  return 0;
}
```

```c
#include <stdio.h>

void Swap(int * x, int * y)
{ int temp = *x;
  *x = *y;
  *y = temp;
}

void BubbleSort(int * x, int n)
{ for(int i=0; i<n-1; i++)
    for (int j=0; j<n-1; j++)
      if(x[j] > x[j+1])     // if adjacent items are not in order
        Swap(&x[j],&x[j+1]); // then swap them!
}
```

```c
int * Search(int target, int * array, int size)
{ int * result = 0;
  for(int i=0;i<size;i++)   // A loop looping n times (n=size)
    if(array[i] == target)  result = &array[i];
  return result;
}

int main(void)
{ int test[5] = {9,8,4,2,1};
  BubbleSort(test,sizeof(test)/sizeof(int));
  printf("%d %d %d %d %d\n",test[0],test[1],test[2],test[3],test[4]);
  int target = 4, * targetPtr = 0;
  targetPtr = Search(target,test,sizeof(test)/sizeof(int));
  return 0;
}
```

```c
#include <stdio.h>
void Swap(int * x, int * y)
{ int temp = *x;
  *x = *y;
  *y = temp;
}

void BubbleSort(int * x, int n)
{ for(int i=0; i<n-1; i++)
    for (int j=0; j<n-1; j++)
      if(x[j] > x[j+1])     // if adjacent items are not in order
        Swap(&x[j],&x[j+1]); // then swap them!
}

int * Search(int target, int * array, int size)
{ int * result = 0;
  for(int i=0;i<size;i++)   // A loop looping n times (n=size)
    if(array[i] == target)  result = &array[i];
  return result;
}

int main(void)
{ int test[5] = {9,8,4,2,1};
  BubbleSort(test,sizeof(test)/sizeof(int));
  printf("%d %d %d %d %d\n",test[0],test[1],test[2],test[3],test[4]);
  int target = 4, * targetPtr = 0;     // Search target is 4
  targetPtr = Search(target,test,sizeof(test)/sizeof(int));
  printf("Result: %d\n",*targetPtr);    // Print out the result if found
  return 0;
}
```

```
C:\Users\Arthur Tang\Documents\test.exe
1 2 4 8 9
Result: 4

--------------------------------
Process exited after 0.01392 seconds with return value 0
Press any key to continue . . .
```

```c
#include <stdio.h>
void Swap(int * x, int * y)
{ int temp = *x;
  *x = *y;
  *y = temp;
}

void BubbleSort(int * x, int n)
{ for(int i=0; i<n-1; i++)
    for (int j=0; j<n-1; j++)
      if(x[j] > x[j+1])     // if adjacent items are not in order
        Swap(&x[j],&x[j+1]); // then swap them!
}

int * Search(int target, int * array, int size)
{ int * result = 0;
  for(int i=0;i<size;i++)   // A loop looping n times (n=size)
    if(array[i] == target)  result = &array[i];
  return result;
}

int main(void)
{ int test[5] = {9,8,4,2,1};
  BubbleSort(test,sizeof(test)/sizeof(int));
  printf("%d %d %d %d %d\n",test[0],test[1],test[2],test[3],test[4]);
  int target = 3, * targetPtr = 0;        // Search target is 3
  targetPtr = Search(target,test,sizeof(test)/sizeof(int));
  if(targetPtr == NULL)   printf("Not Found!!!\n");
  else                    printf("Result: %d\n",*targetPtr);
  return 0;
}
```

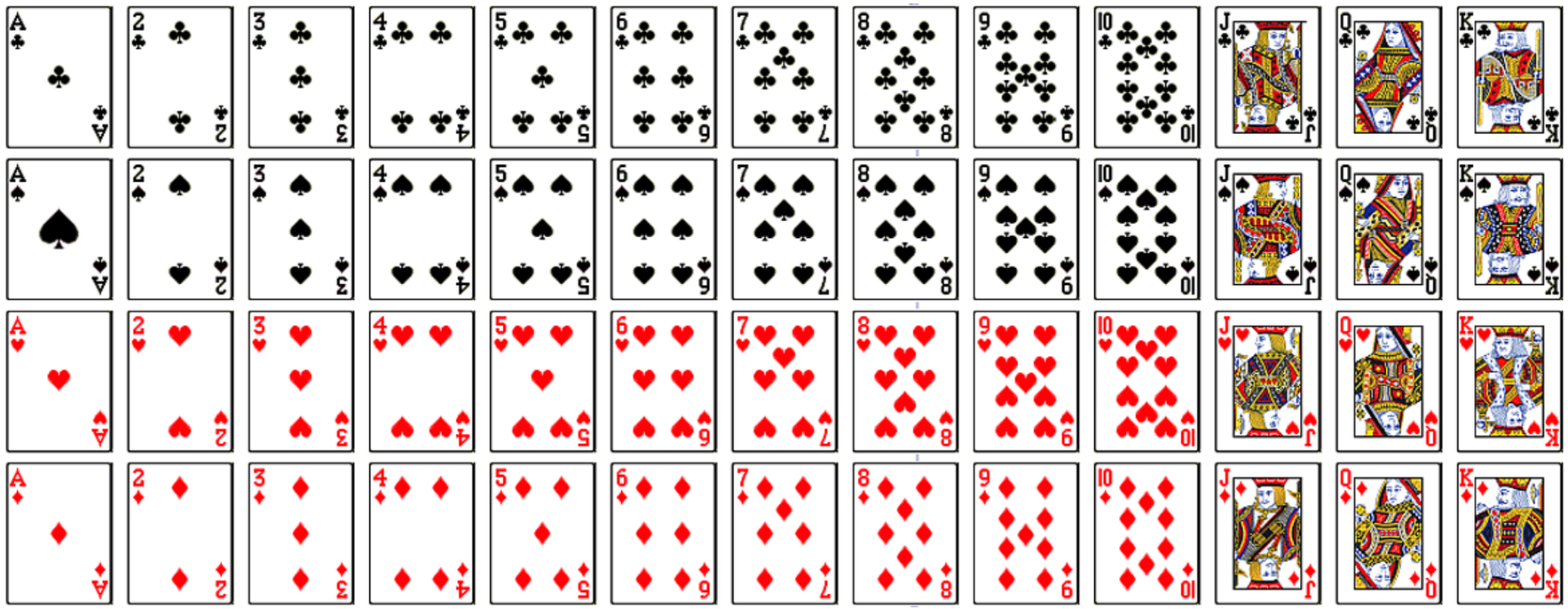
C:\Users\Arthur Tang\Documents\test.exe

```
1 2 4 8 9
Not Found!!!

--------------------------------
Process exited after 0.01905 seconds with return value 0
Press any key to continue . . .
```

Another Pointer Example: Deck of Cards

- We want to write a simulator for standard deck of card
- The simulator will shuffle the 52 cards randomly
- Then you can use the program to deal the card to players (according to your game rule)
- How do we do this???

Out[26]=



Data Structure:
How do you represent 52 cards

- We use 4-by-13 double-subscripted array `deck` to represent the deck of playing cards.
- The rows correspond to the *suits*—row 0 corresponds to hearts, row 1 to diamonds, row 2 to clubs and row 3 to spades.
- The columns correspond to the *face* values of the cards—0 through 9 correspond to ace through ten, and columns 10 through 12 correspond to jack, queen and king.

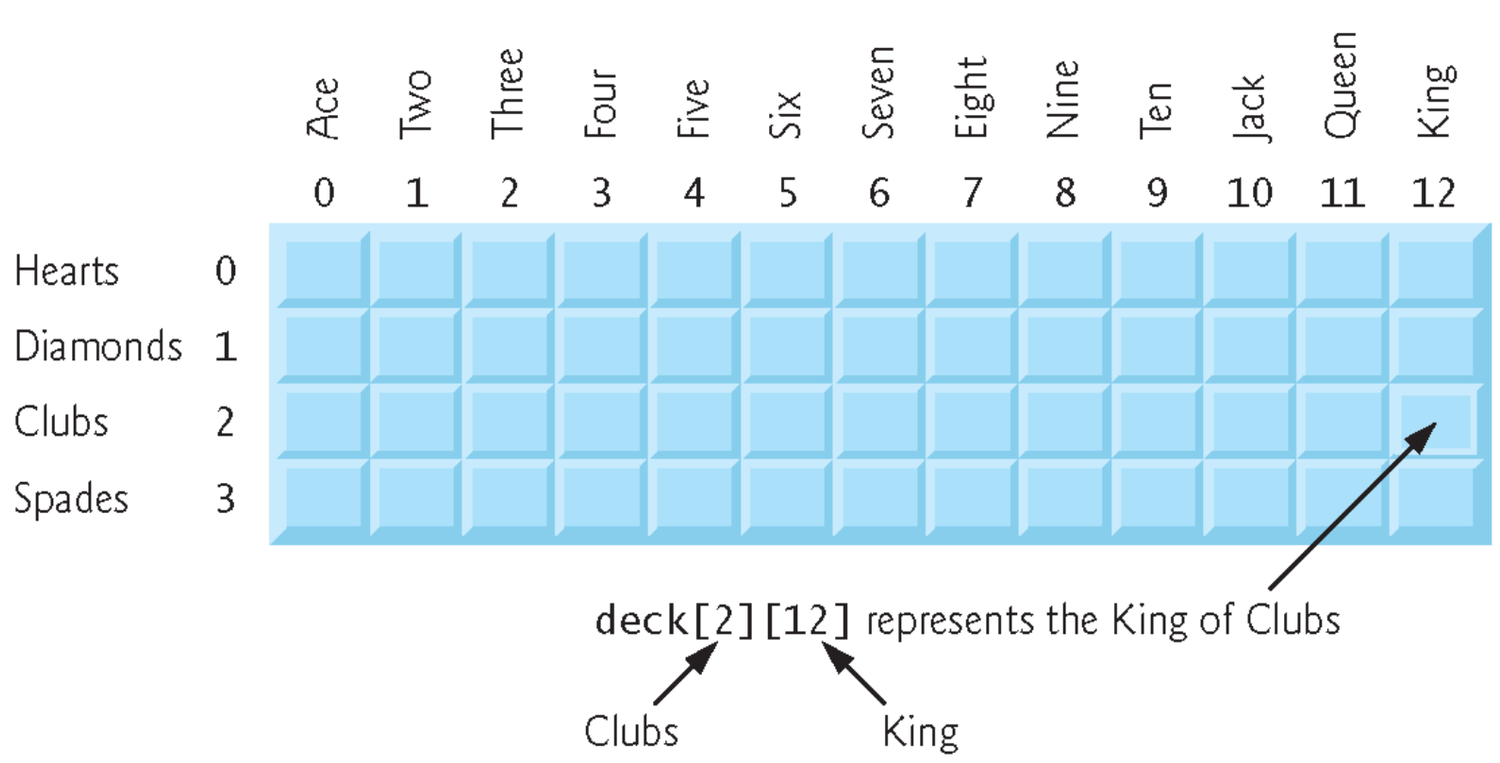


Fig. 7.23 | Double-subscripted array representation of a deck of cards.

Data Structure:
How do you represent 52 cards

- We can then create a 13 x 4 2 dimensional array
  - `int deck[4][13] = {0};`
- In this table, the number represent the order of the card on the deck, for example:

	A	2	3	4	5	6	7	8	9	10	J	Q	K
Spades	16	30	9	35	3	22	10	4	23	32	33	26	37
Hearts	31	1	15	36	2	21	6	11	34	40	24	25	48
Diamonds	17	52	8	14	20	7	39	28	5	27	44	43	47
Clubs	42	18	46	19	13	38	12	29	41	45	49	50	51

- The first card on the deck is 2 of Hearts
- The second card on the deck is 5 of Hearts
- The third card on the deck is 5 of Spades
- Etc … … … ...

Data Structure: Shuffling the cards

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>


void shuffle(int deck[][13])
{ // How do you shuffle the deck???
}
```

```
int main(void)
{ int deck[4][13] = {0};
  srand(time(NULL));
  shuffle(deck);
}
```

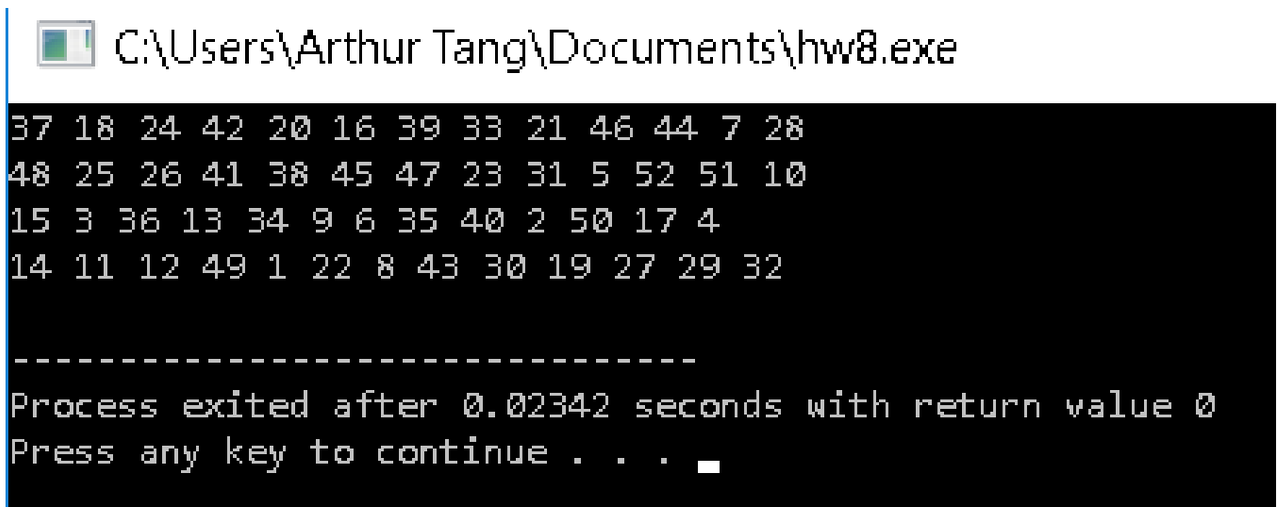
Data Structure: Shuffling the cards

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>


void shuffle(int deck[][13])
{ for(int i=0; i< 52; i++) // 52 cards for 1 deck
   {
   }
}

int main(void)
{ int deck[4][13] = {0};
   srand(time(NULL));
   shuffle(deck);
}
```

Data Structure: Shuffling the cards

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void shuffle(int deck[][13])
{ for(int i=1; i<=52; i++) // 52 cards for 1 deck
  { int num;
    do {
      num = rand()%52;      // Select a random number from 0-51
    }while(deck[num/13][num%13] != 0); // If the table is not empty, select anothe
    deck[num/13][num%13] = i;          // Set Table entry to the value of i
  }
}

int main(void)
{ int deck[4][13] = {0};
  srand(time(NULL));
  shuffle(deck);
}
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
void shuffle(int deck[][13])
{ for(int i=1; i<=52; i++) // 52 cards for 1 deck
  { int num;
    do {
      num = rand()%52;      // Select a random number from 0-51
    }while(deck[num/13][num%13] != 0); // If the table is not empty, select another
    deck[num/13][num%13] = i;          // Set Table entry to the value of i
  }
}

int main(void)
{ int deck[4][13] = {0};
  srand(time(NULL));
  shuffle(deck);
  for(int i=0;i<4;i++)     // Print out the shuffled deck
  { for(int j=0;j<13;j++)
      printf("%d ",deck[i][j]);
    printf("\n");
  }
}
```



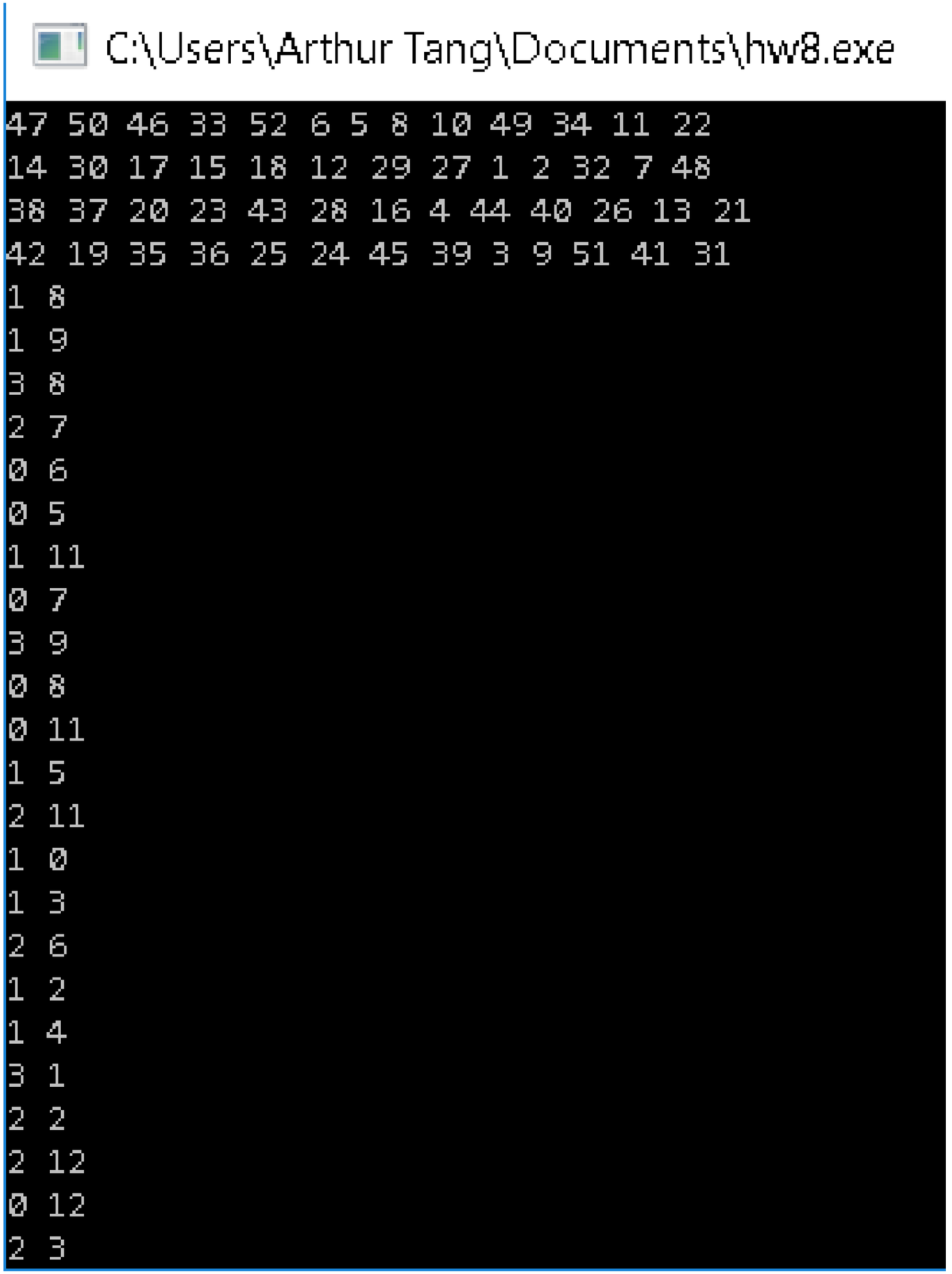
```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int sequence[52][2] = {0};// An array storing the sequence of card for the deck

void shuffle(int deck[][13])
{ for(int i=1; i<=52; i++) // 52 cards for 1 deck
  { int num;
    do {
      num = rand()%52;      // Select a random number from 0-51
    }while(deck[num/13][num%13] != 0); // If the table is not empty, select another
    deck[num/13][num%13] = i;          // Set Table entry to the value of i
  }
}
```

```
int main(void)
{ int deck[4][13] = {0};
  srand(time(NULL));
  shuffle(deck);
}
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int sequence[52][2] = {0};// An array storing the sequence of card for the deck

void shuffle(int deck[][13])
{ for(int i=1; i<=52; i++) // 52 cards for 1 deck
  { int num;
    do {
      num = rand()%52;      // Select a random number from 0-51
    }while(deck[num/13][num%13] != 0); // If the table is not empty, select another
    deck[num/13][num%13] = i;          // Set Table entry to the value of I
    sequence[i-1][0] = num/13;         // Store the suit of the current card
    sequence[i-1][1] = num%13;         // Store the face of the current card
  }
}

int main(void)
{ int deck[4][13] = {0};
  srand(time(NULL));
  shuffle(deck);
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int sequence[52][2] = {0};// An array storing the sequence of card for the deck

void shuffle(int deck[][13])
{ for(int i=1; i<=52; i++) // 52 cards for 1 deck
  { int num;
    do {
      num = rand()%52;      // Select a random number from 0-51
    }while(deck[num/13][num%13] != 0); // If the table is not empty, select another
    deck[num/13][num%13] = i;          // Set Table entry to the value of I
    sequence[i-1][0] = num/13;         // Store the suit of the current card
    sequence[i-1][1] = num%13;         // Store the face of the current card
  }
}
```

```
int main(void)
{ int deck[4][13] = {0};
  srand(time(NULL));
  shuffle(deck);
  for (int i=0;i<52;i++)                    // Print out the sequence of card
    printf("%d %d\n",sequence[i][0],sequence[i][1]);
}
```


C:\Users\Arthur Tang\Documents\hw8.exe
47 50 46 33 52 6 5 8 10 49 34 11 22
14 30 17 15 18 12 29 27 1 2 32 7 48
38 37 20 23 43 28 16 4 44 40 26 13 21
42 19 35 36 25 24 45 39 3 9 51 41 31
1 8
1 9
3 8
2 7
0 6
0 5
1 11
0 7
3 9
0 8
0 11
1 5
2 11
1 0
1 3
2 6
1 2
1 4
3 1
2 2
2 12
0 12
2 3

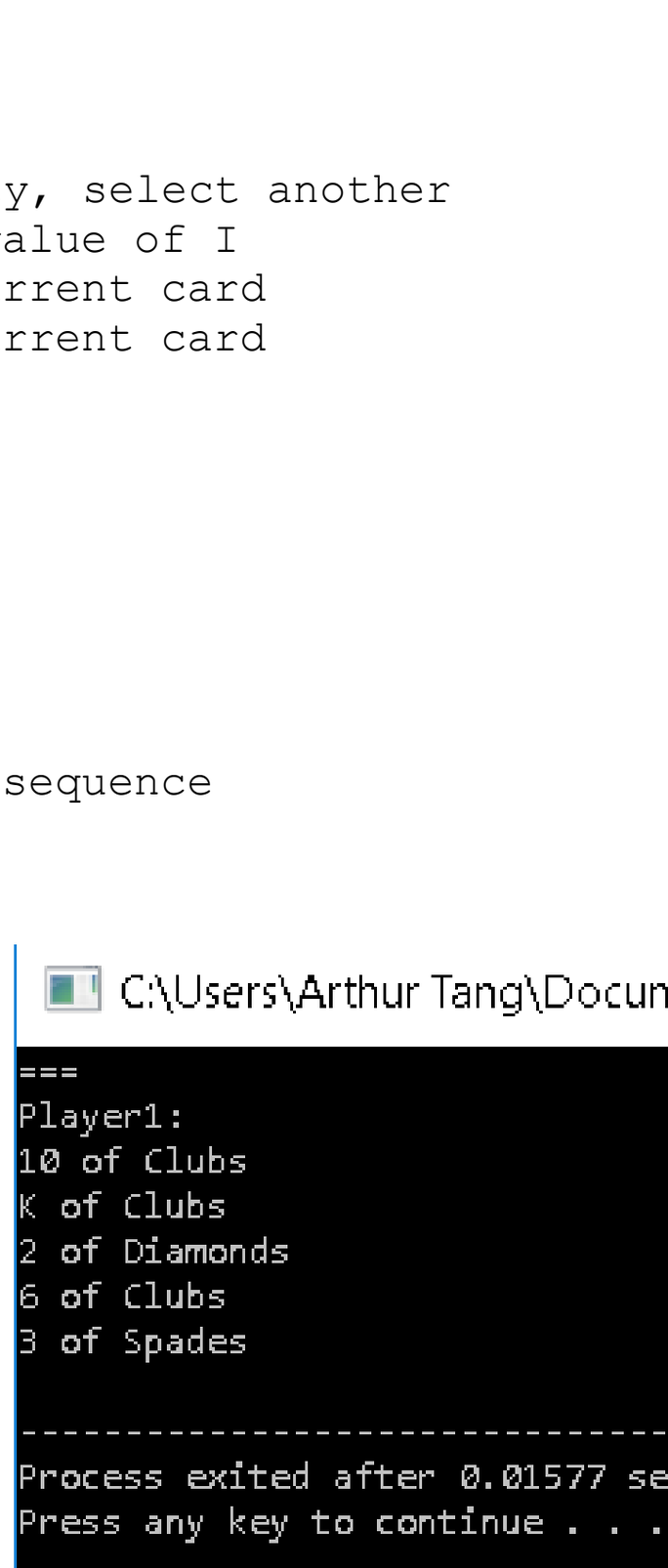
```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int sequence[52][2] = {0};   // An array storing the sequence of card for the deck

void shuffle(int deck[][13])
{ for(int i=1; i<=52; i++) // 52 cards for 1 deck
  { int num;
    do {
      num = rand()%52;     // Select a random number from 0-51
    }while(deck[num/13][num%13] != 0); // If the table is not empty, select another
    deck[num/13][num%13] = i;          // Set Table entry to the value of I
    sequence[i-1][0] = num/13;         // Store the suit of the current card
    sequence[i-1][1] = num%13;         // Store the face of the current card
  }
}

int main(void)
{ int deck[4][13] = {0};
  srand(time(NULL));
  shuffle(deck);
  for (int i=0;i<52;i++)              // Print out the name of the card
  { if(sequence[i][1] == 0)          printf("A");
    else if(sequence[i][1] == 10)    printf("J");
    else if(sequence[i][1] == 11)    printf("Q");
    else if(sequence[i][1] == 12)    printf("K");
    else                             printf("%d",sequence[i][1]+1);

    if(sequence[i][0] == 0)          printf(" of Spades\n");
    else if(sequence[i][0] == 1)     printf(" of Hearts\n");
    else if(sequence[i][0] == 2)     printf(" of Diamonds\n");
    else if(sequence[i][0] == 3)     printf(" of Clubs\n");
  }
}
```

C:\Users\Arthur Tang\Documents\hw8.exe

```
29 27 1 25 37 46 33 42 24 26 8 39 4
34 11 50 40 13 48 6 51 32 49 10 41 17
36 19 18 22 30 31 28 23 9 20 52 21 5
43 14 47 15 3 7 44 38 45 12 35 16 2
3 of Spades
K of Clubs
5 of Clubs
K of Spades
K of Diamonds
7 of Hearts
6 of Clubs
J of Spades
9 of Diamonds
J of Hearts
2 of Hearts
10 of Clubs
5 of Hearts
2 of Clubs
4 of Clubs
Q of Clubs
K of Hearts
3 of Diamonds
2 of Diamonds
10 of Diamonds
Q of Diamonds
4 of Diamonds
8 of Diamonds
```

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int sequence[52][2] = {0};   // An array storing the sequence of card for the deck

void shuffle(int deck[][13])
{ for(int i=1; i<=52; i++) // 52 cards for 1 deck
  { int num;
    do {
      num = rand()%52;     // Select a random number from 0-51
    }while(deck[num/13][num%13] != 0); // If the table is not empty, select another
    deck[num/13][num%13] = i;          // Set Table entry to the value of I
    sequence[i-1][0] = num/13;         // Store the suit of the current card
    sequence[i-1][1] = num%13;         // Store the face of the current card
  }
}

int main(void)
{ int deck[4][13] = {0};
  srand(time(NULL));
  shuffle(deck);
  int * Player1[5];
  for(int i=0;i<5;i++)           // Pass 5 cards to each player in sequence
    Player1[i] = &sequence[i][0];
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int sequence[52][2] = {0};   // An array storing the sequence of card for the deck

void shuffle(int deck[][13])
{ for(int i=1; i<=52; i++) // 52 cards for 1 deck
  { int num;
    do {
      num = rand()%52;     // Select a random number from 0-51
    }while(deck[num/13][num%13] != 0); // If the table is not empty, select another
    deck[num/13][num%13] = i;          // Set Table entry to the value of I
    sequence[i-1][0] = num/13;         // Store the suit of the current card
    sequence[i-1][1] = num%13;         // Store the face of the current card
  }
}

int main(void)
{ int deck[4][13] = {0};
  srand(time(NULL));
  shuffle(deck);
  int * Player1[5];
  for(int i=0;i<5;i++)           // Pass 5 cards to each player in sequence
    Player1[i] = &sequence[i][0];

  printf("===\nPlayer1: \n");   // Print out the cards of Player1
  for (int i=0;i<5;i++)
  { if(*(Player1[i]+1) == 0)       printf("A");
    else if(*(Player1[i]+1) == 10) printf("J");
    else if(*(Player1[i]+1) == 11) printf("Q");
    else if(*(Player1[i]+1) == 12) printf("K");
    else                           printf("%d",*(Player1[i]+1)+1);

    if(*Player1[i] == 0)       printf(" of Spades\n");
    else if(*Player1[i] == 1)  printf(" of Hearts\n");
    else if(*Player1[i] == 2)  printf(" of Diamonds\n");
    else if(*Player1[i] == 3)  printf(" of Clubs\n");
  }
```



Q&A?