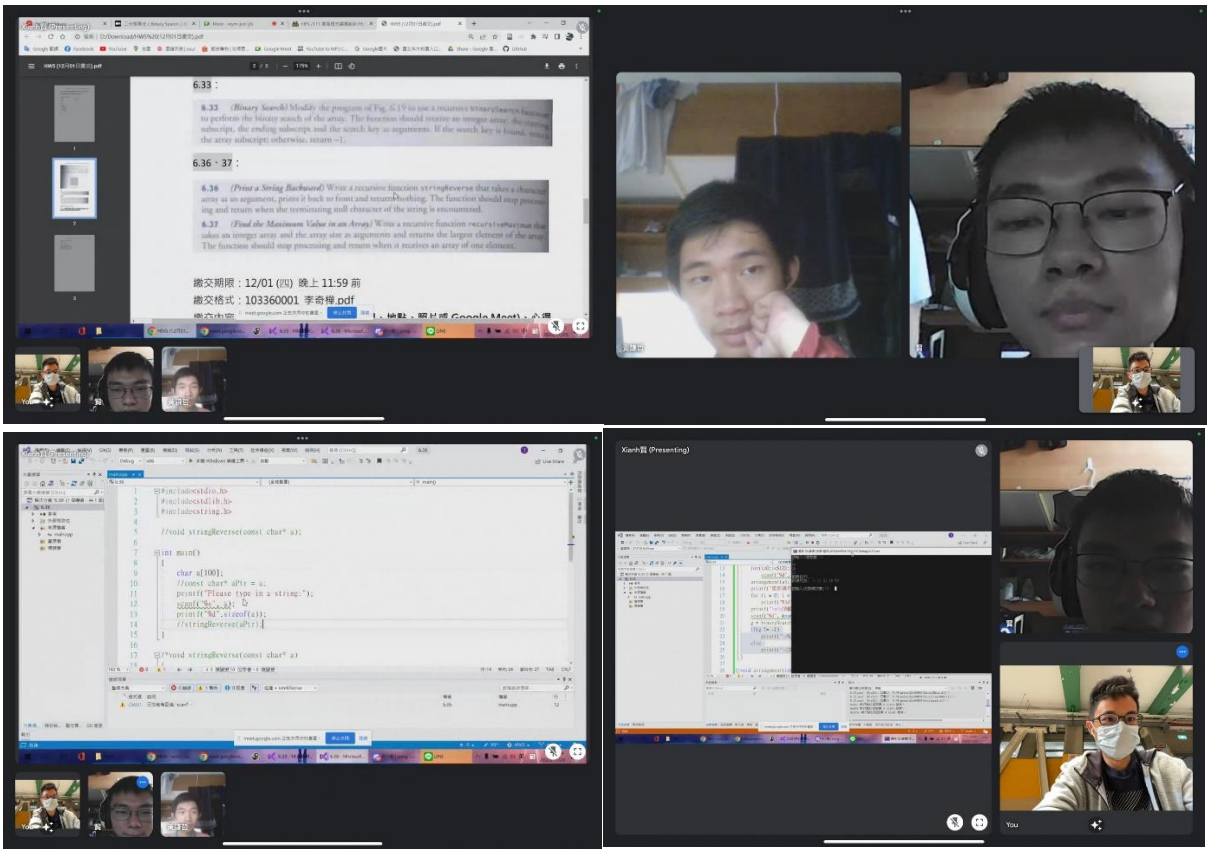
Discussion Time: 28th November 2020, 15:03 - 15:17



P05

main.c

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

int main()
{
    int a;
    int *aPtr;

    a = 7;
    aPtr = &a;

    printf("The address of a is %p\nThe value of aPtr is %p\n", &a, aPtr);
    printf("\nThe value of a is %d\nThe value of *aPtr is %d\n", a, *aPtr);
    printf("\nShowing that * and & are complements of each other\n"
           "&*aPtr = %p\n*&aPtr = %p\n", &*aPtr, *&aPtr);

    system ("pause");
    return 0;
}
```

P06

main.c

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

int cubeValue(int n);

int main()
{
    int num = 5;
    printf("The original value of num is: %d\n", num);

    num = cubeValue(num);
    printf("The modified value of num is: %d\n", num);

    system ("pause");
    return 0;
}

int cubeValue(int n) {
    return n * n * n;
}
```

P10

main.c

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

void cubeAddress(int *n);

int main()
{
    int num = 5;
    printf("The original value of num is: %d\n", num);

    cubeAddress(&num);
    printf("The modified value of num is: %d\n", num);

    system ("pause");
    return 0;
}

void cubeAddress(int *n) {
    *n = *n * *n * *n;
}
```

P13

main.cpp

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

void cubeReference(int &n);

int main()
{
    int num = 5;
    printf("The original value of num is: %d\n", num);

    cubeReference(num);
    printf("The modified value of num is: %d\n", num);

    system ("pause");
    return 0;
}

void cubeReference(int &n) {
    n = n * n * n;
}
```

P15

main.c

```c
#include <stdio.h>
#include <stdlib.h>
#define Size 10

void bubbleSort(int *const array, const int size);

int main()
{
    int a[Size] = {2,6,4,8,10,12,89,68,45,37};
    int i;

    printf("Data items in original order\n");
    for (i = 0; i < Size; i++) {
        printf("%4d", a[i]);
    }

    bubbleSort(a, Size);

    printf("\nData items in ascending order\n");
    for (i = 0; i < Size; i++) {
        printf("%4d", a[i]);
    }
    printf("\n");

    system ("pause");
    return 0;
}

void bubbleSort(int * const array, const int size) {
    void swap(int * element1, int * element2);
    int i, j;

    for(i = 0; i < size-1; i++) {
        for(j = 0; j < size-1; j++) {
            if (array[j] > array[j+1]) {
                swap(&array[j], &array[j+1]);
            }
        }
    }
}

void swap(int *element1, int *element2) {
    int temp = *element1;
    *element1 = *element2;
    *element2 = temp;
}
```

P19

main.c

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

int main()
{
    int b[] = {10,20,30,40};
    int *bPtr = b;
    int i;
    int offset;

    printf("Array b printed with:\nArray subscipt notation\n");
    for (i = 0; i < 4; i++) {
        printf("b[%d] = %d\n", i, b[i]);
    }

    printf("\nPointer/offset notation where the pointer is in the array name\n");
    for (offset = 0; offset < 4; offset++) {
        printf("*(b + %d) = %d\n", offset, *(b+offset));
    }

    printf("\nPointer subscript notation\n");
    for (i = 0; i < 4; i++) {
        printf("bPtr[%d] = %d\n", i, bPtr[i]);
    }

    printf("\nPointer/offset notation\n");
    for (offset = 0; offset < 4; offset++) {
        printf("*(bPtr + %d) = %d\n", offset, *(bPtr+offset));
    }

    system ("pause");
    return 0;
}
```

P25

main.c

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

void shuffle(int leDeck[][13]);
void deal(const int leDeck[][13], const char *leFace[], const char *leSuit[]);

int main()
{
    const char *suit[4] = {"Hearts", "Diamonds", "Clubs", "Spades"};
    const char *face[13] = {"Ace", "One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", "Nine", "Jack", "Queen", "King"};

    int deck[4][13] = {0};

    srand(time(0));

    shuffle(deck);
    deal(deck, face, suit);

    system ("pause");
    return 0;
}

void shuffle(int leDeck[][13]) {
    int row, column, card;

    for(card = 1; card <= 52; card++) {
        do {
            row = rand() % 4;
            column = rand() % 13;
        } while (leDeck[row][column] != 0);

        leDeck[row][column] = card;
    }
}

void deal(const int leDeck[][13], const char *leFace[], const char *leSuit[]) {
    int row, column, card;

    for (card = 1; card <= 52; card++) {
        for(row = 0; row <= 3; row++) {
            for (column = 0; column <= 12; column++) {
                if (leDeck[row][column] == card) {
                    printf("%5s of %-8s%c", leFace[column], leSuit[row], card % 2 == 0 ? '\n' : '\t');
                }
            }
        }
    }
}
```

6.19

main.c

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

void diceRoll(int roll1, int roll2, int counter[]);
void frequencyPrinter(int counter[]);

int main()
{
    int dice1, dice2, i;
    int rolled[13] = {0};

    srand(time(NULL));

    diceRoll(dice1, dice2, &rolled);
    frequencyPrinter(rolled);

    system ("pause");
    return 0;
}

void diceRoll(int roll1, int roll2, int counter[]) {
    int j;
    for (j = 1; j <= 36000; j++) {
        roll1 = 1 + rand()%6;
        roll2 = 1 + rand()%6;
        counter[roll1 + roll2]++;
    }
}

void frequencyPrinter(int counter[]) {
    int i;

    for (i = 2; i < 13; i++) {
        printf("%5d%12d\n", i, counter[i]);
    }
}
```

6.33

main.c

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

int binarySearch(int array[], int left, int right, int key);

int main()
{
    int a[8] = { 6,16,34,56,57,79,86,98 };
    int key, result, size;

    printf("Input key: "); scanf("%d", &key);
    size = sizeof(a)/sizeof(a[0]);

    result = binarySearch(a, 0, size-1, key);

    if (result == -1)
        printf("Element not found within array\n");
    else
        printf("Element found at %d\n", result);

    system ("pause");
    return 0;
}

int binarySearch(int array[], int left, int right, int key) {
    int mid;

    if (right >= left) {
        mid = left + (right - left)/2;

        if (array[mid] == key)
            return mid;
        if (array[mid] > key)
            return binarySearch(array, left, mid-1, key);
        else
            return binarySearch(array, mid+1, right, key);
    }
    else
        return -1;
}
```

6.36

main.c

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

void reverseEm(char *word);

int main()
{
    char sheesh[] = "";
    printf("Input string: "); scanf("%s", &sheesh);
    reverseEm(sheesh);
    printf("\n");

    system ("pause");
    return 0;
}

void reverseEm(char *word) {
    if (*word != '\0') {
        reverseEm(word + 1);
        printf("%c", *word);
    }
}
```

6.37

main.c

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

void max(int *num, int size, int maximum);

int main()
{
    int a[5] = { 195,234,154,745,256 };
    int size = sizeof(a)/sizeof(a[0]);
    max(a, size-1, a[0]);

    system ("pause");
    return 0;
}

void max(int *num, int size, int maximum) {
    if (size < 0) {
        printf("The maximum element within array is: %d", maximum);
    }
    else {
        if (*num > maximum)
            maximum = *num;

        max(num+1, size-1, maximum);
    }
}
```

Conclusion:

The usage of pointers can simplify a program, maybe even allow certain programs to work. But the usage of pointers can make yourself confusing. Recursive function can also simplify a program, where the function calls itself. But using recursive can make one become extremely confused. For pointers, we can assign them using '*', and address can be assigned using '&'. Recursive is simply a function containing itself, though one needs to remember that it will become a loop, meaning it needs a terminating condition.

Code: https://github.com/AldrichWijaya/Homework.git