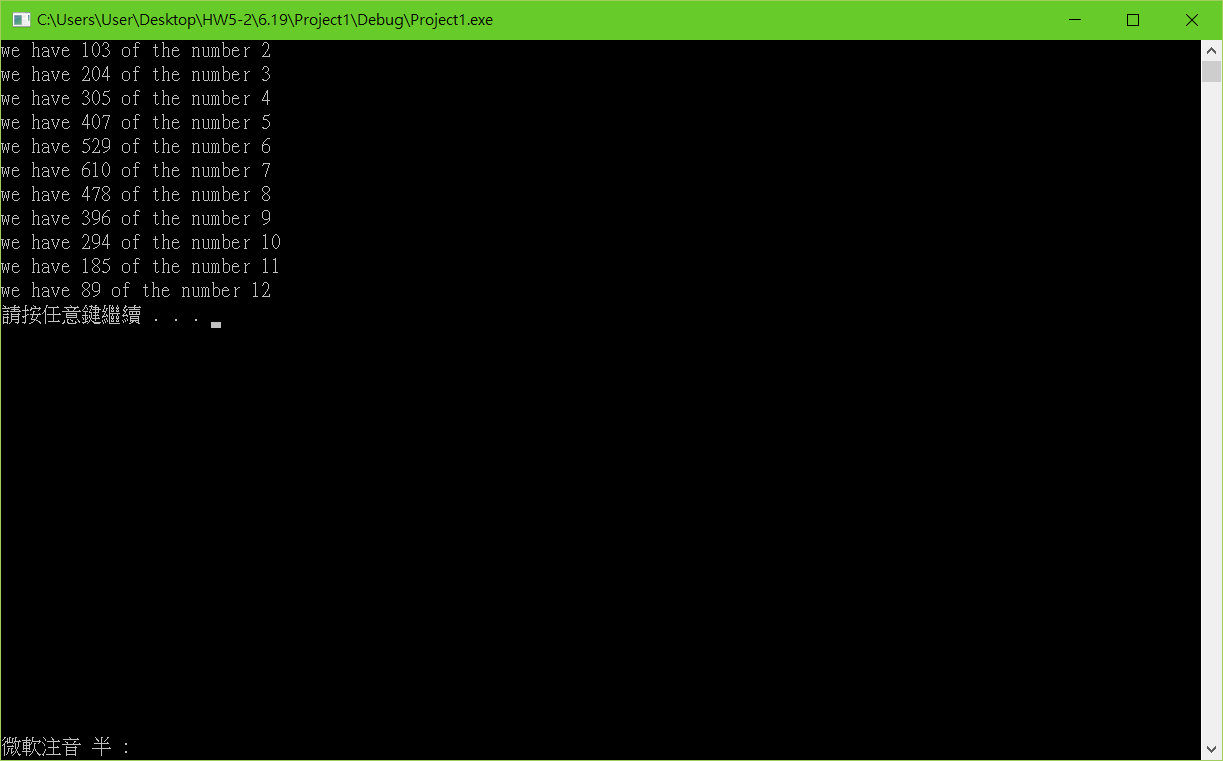
**6.19**



#include <stdio.h>

#include <stdlib.h>

#include <time.h>

void shuffle(int wDeck[][6]);

int main(void)

{

const int \*suit[6] = { 1,2,3,4,5,6 };

const int \*face[6] ={ 1,2,3,4,5,6 };

int deck[6][6] = { {2,3,4,5,6,7} ,{3,4,5,6,7,8},{4,5,6,7,8,9},

{5,6,7,8,9,10},{6,7,8,9,10,11},{7,8,9,10,11,12} };

srand(time(0));

shuffle(deck);

system("pause");

return 0;

}

void shuffle(int wDeck[][6])

{

int row, column, i;

int D[11] = { 0 };

for (i = 1; i <= 3600; i++)

{

row = rand() % 6;

column = rand() % 6;

switch (wDeck[row][column])

{

case 2:

D[0] = D[0] + 1;

break;

case 3:

D[1] = D[1] + 1;

break;

case 4:

D[2] = D[2] + 1;

break;

case 5:

D[3] = D[3] + 1;

break;

case 6:

D[4] = D[4] + 1;

break;

case 7:

D[5] = D[5] + 1;

break;

case 8:

D[6] = D[6] + 1;

break;

case 9:

D[7] = D[7] + 1;

break;

case 10:

D[8] = D[8] + 1;

break;

case 11:

D[9] = D[9] + 1;

break;

case 12:

D[10] = D[10] + 1;

break;

default:

break;

}

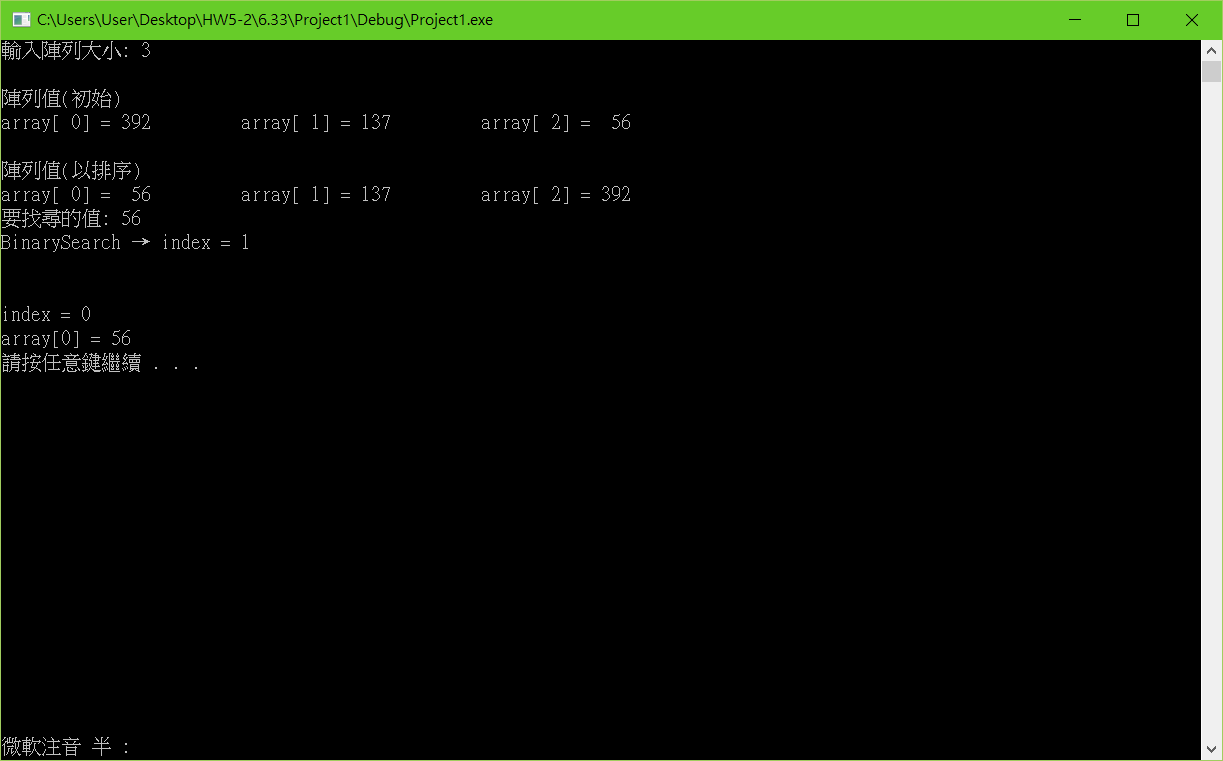
}

for ( i = 0; i <= 10; i++)

printf("we have %d of the number %d\n", D[i], i + 2);

}

**6.33**



#include <stdio.h>

#include <stdlib.h>

void printArray(int \*array) {

int n = 0;

while (array[n] != NULL) n++;

for (int i = 0; i < n; i++) {

if (!(i % 5)) printf("\n");

printf("array[%2d] = %3d \t", i, array[i]);

}

printf("\n");

}

int cmpfunc(const void \*a, const void \*b) {

return (\*(int\*)a - \*(int\*)b);

}

int BinarySearch(int value, int \*array, int start, int end) {

int index = (start + end) / 2;

if (start == end)return (value == array[index]) ? index : -1;

printf("BinarySearch → index = %d\n", index);

if (value == array[index]) return index;

else if (value > array[index]) return BinarySearch(value, array, index + 1, end);

else if (value < array[index]) return BinarySearch(value, array, start, (index > 0) ? index - 1 : index);

}

int main() {

int \*array, size;

srand(time(NULL));

printf("輸入陣列大小: ");

scanf\_s("%d", &size);

array = calloc(size, sizeof(int));

\*(array + size) = NULL;

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

array[i] = rand() % 500;

printf("\n陣列值(初始)");

printArray(array);

qsort(array, size, sizeof(int), cmpfunc);

printf("\n陣列值(以排序)");

printArray(array);

printf("要找尋的值: ");

int value, index;

scanf\_s("%d", &value);

index = BinarySearch(value, array, 0, size);

printf("\n\nindex = %d\n", index);

if (index == -1)

printf("無法在陣列中找到 %d\n", value);

else

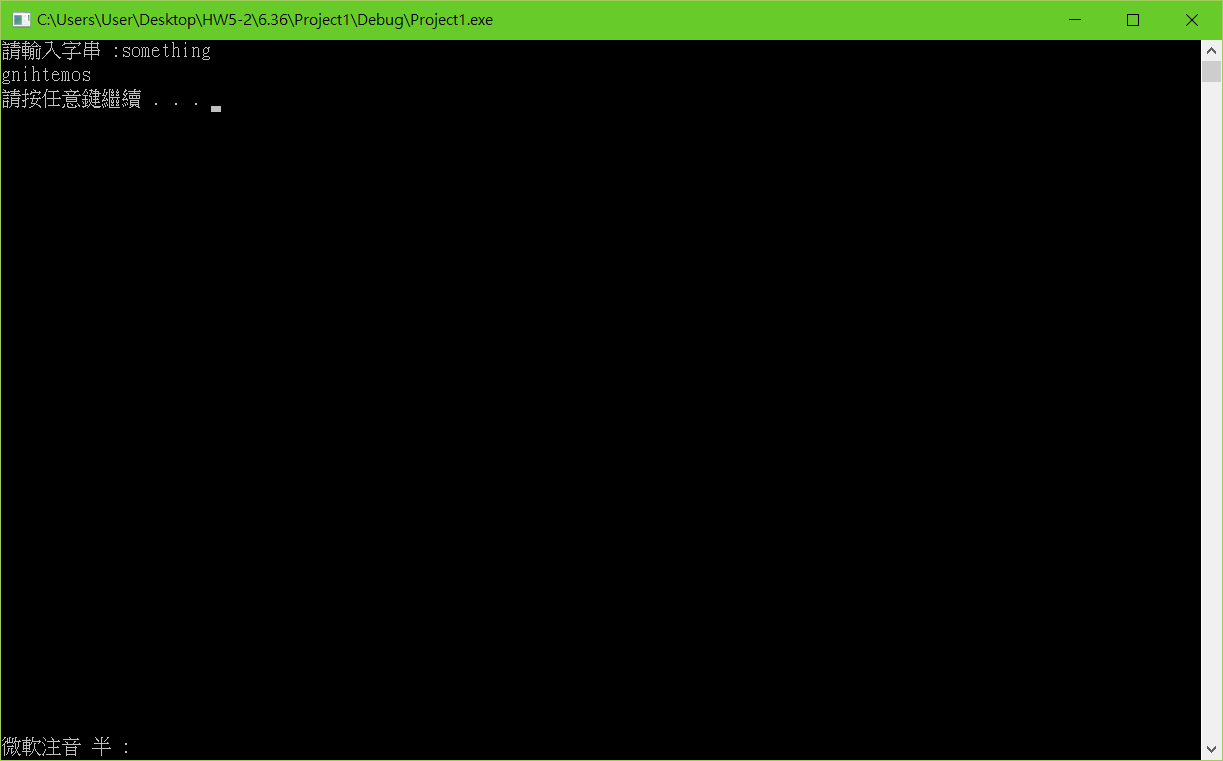
printf("array[%d] = %d\n", index, value);

system("pause");

return 0;

}

**6.36**



#include <stdio.h>

#include <stdlib.h>

void printStringBackward(char \*str, int i) {

if (str[i] != '\0') {

printStringBackward(str, i + 1);

printf("%c", str[i]);

}

}

int main(void) {

char str[60];

printf("請輸入字串 :");

gets(str);

printStringBackward(str, 0);

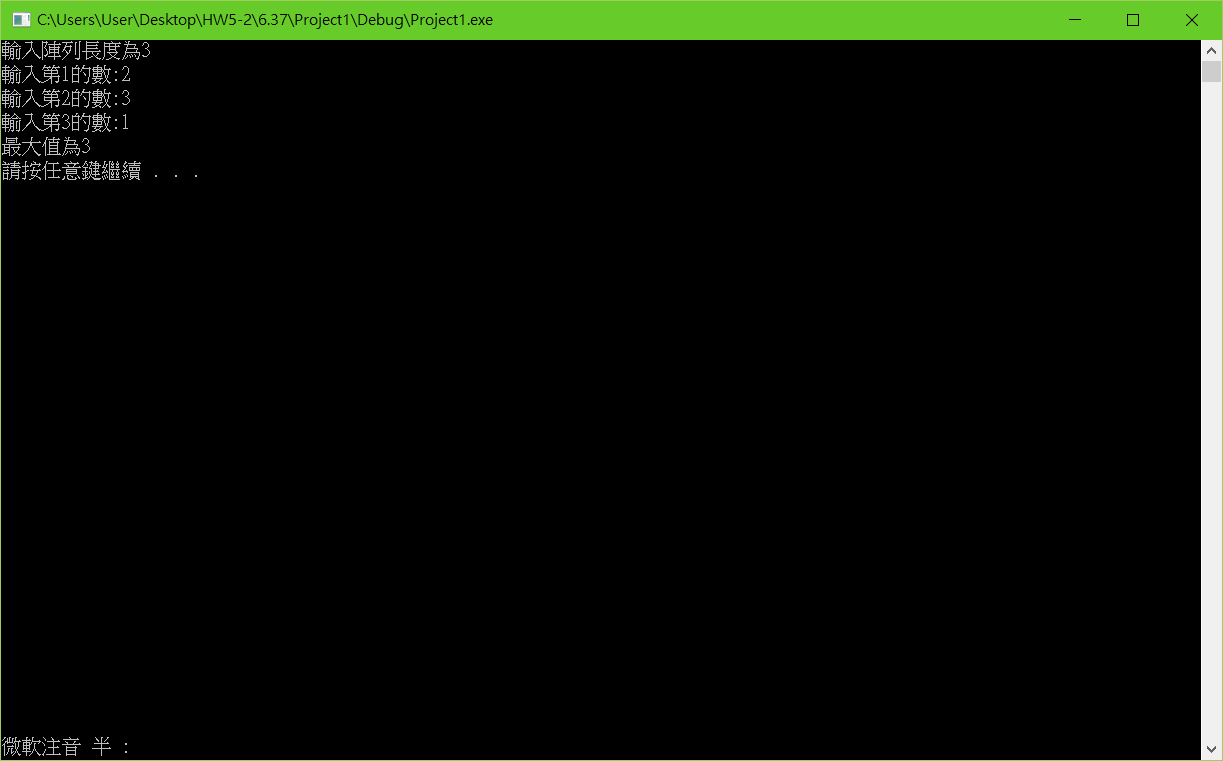
printf("\n");

system("pause");

return 0;

}

**6.37**



#include <stdio.h>

#include <stdlib.h>

int recursiveMaximum(int num[], int u);

int u = 3;

int main()

{

int i, ans, n;

printf("輸入陣列長度為3\n");

int x[3];

for (i = 0; i < 3; i++)

{

printf("輸入第%d的數:", i + 1);

scanf("%d", &n);

x[i] = n;

}

ans = recursiveMaximum(x, u);

printf("最大值為%d", ans);

printf("\n");

system("pause");

return 0;

}

int recursiveMaximum(int num[], int u)

{

if (u == 1)

return num[0];

if (recursiveMaximum(num, u - 1) < num[u - 1])

return num[u - 1];

else

return recursiveMaximum(num, u - 1);

recursiveMaximum(num, u - 1);

}

**Git**

**心得:**

**這次的程式實在很難，再跟同學討論後才能夠了解如何完成。**