DSA Homework 2

B03902089 林良翰

**2.1 (1)**

The variable ‘c’ will be deleted by default when the function ‘sub1’ is returned. As a result, there would be run time error after executing ‘sub1’ because the variable ‘c’ isn’t exist,

**2.1 (2)**

The problem is that we have to delete the variable ‘pc’, or it will cause memory leak after the whole program closed.

**2.2 (1)**

bool repeat(int i, int B[], int max, int pos, int times) {

if (times >= 2) return true;

else if (pos >= max) return false;

else if (i == B[pos]) {

return repeat(i, B, max, pos+1, times+1);

} else {

pos++;

return repeat(i, B, max, pos+1, times);

}

}

bool find(int B[]) {

int repeated = 0;

for (int i = 1 ; i <= n-5 ; i++) {

if ( repeat(i, B, n, 0, 0)) {

repeated++;

}

if ( repeated == 5) return true;

}

return false;

}

**2.2 (2)**

**Get from the matrix :**

int get(int array[], int row, int column) {

int n = 0;

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

return array[n+column];

}

**Put to the matrix :**

void put(int array[], int row, int column, int input) {

int n = 0;

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

array[n+column] = input;

}

**Memory layout :**

**2.2 (3)**

class LinkList {

public:

//data...

LinkList \*next;

};

bool isSame(LinkList \*L\_head, LinkList \*M\_head) {

LinkList \*start\_L = L\_head;

LinkList \*start\_M = M\_head;

int headsame = 0;

do {

if (start\_L->data == M\_head->data) {

headsame = 1;

break;

}

start\_L = start\_L->next;

} while (start\_L =! L\_head);

if (headsame) {

do {

if (start\_L->data != start\_M->data) return false;

start\_L = start\_L->next;

start\_M = start\_M->next;

} while (start\_L != L\_head);

return true;

} else return false;

}

**2.2 (4)**

void swap(int \*a, int \*b) {

int temp = \*a;

\*a = \*b;

\*b = temp;

}

void rearrange(int array[], int pos, int swapPos, int max) {

if (pos >= max || swapPos >= max) return;

else if (array[pos]%2 != 0) {

swap(&array[pos], &array[swapPos]);

rearrange(array, pos, swapPos+1, max);

} else

rearrange(array, pos+1, pos+2, max);

}

int main() {

int array[max];

//....process array

rearrange(array, 0, 1, max);

return 0;

}

**2.2 (5)**

int swapPos = 1;

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

if (array[i]%2 == 0) {

swapPos = i + 2;

continue;

} else if (array[i]%2 != 2) {

swap(array[i], array[swapPos]);

swapPos++;

}

}

2.3