# Contents

# 1 sort

## 1.1 sort number

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

int values[] = { 88, 56, 100, 2, 25 };

int cmpfunc (const void * a, const void * b)
{
  return ( *(int*)a - *(int*)b );
}

// 2 25 56 88 100
qsort(values, 5, sizeof(int), cmpfunc);
```

## 1.2 sort string

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

int compare(const void *a, const void *b) {
  return strcmp(*(const char **)a, *(const char **)b);
}

const char *arr[]
    = {"apple", "orange", "banana", "grape", "cherry"};

int n = sizeof(arr) / sizeof(arr[0]);

// apple banana cherry grape orange
qsort(arr, n, sizeof(const char *), compare);
```

## 1.3 bubble sort

```
#include <stdio.h>

void swap(int *xp, int *yp)
{
  int temp = *xp;
  *xp = *yp;
  *yp = temp;
}

void bubbleSort(int arr[], int n)
{
  int i, j;
  for (i = 0; i < n - 1; i++)
  {
    for (j = 0; j < n - i - 1; j++)
    {
      if (arr[j] > arr[j + 1])
      {
        swap(&arr[j], &arr[j + 1]);
      }
    }
  }
}
```

# 2 DataStructure

## 2.1 BST

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

struct node
{
  int val;
  struct node *left, *right;
} typedef node;

node *insert(node *now, int val)
{
  if (now == NULL)
  {
    node *newnode = (node *)malloc(sizeof(node));
    newnode->val = val;
    newnode->left = newnode->right = NULL;
    return newnode;
  }
  if (now->val > val)
  {
    now->left = insert(now->left, val);
  }
  else if (now->val < val)
  {
    now->right = insert(now->right, val);
  }
  return now;
}

node *deletenode(node *now, int val)
{
  if (now == NULL)
    return now;
  if (now->val > val)
  {
    now->left = deletenode(now->left, val);
  }
  else if (now->val < val)
  {
    now->right = deletenode(now->right, val);
  }
  else
  {
    if (now->left == NULL)
    {
      node *tmp = now->right;
      free(now);
      return tmp;
    }
    else if (now->right == NULL)
    {
      node *tmp = now->left;
      free(now);
      return tmp;
    }
    else
    {
      node *tmp = now->right;
      while (tmp->left)
        tmp = tmp->left;
      now->val = tmp->val;
      now->right = deletenode(now->right, now->val);
    }
  }
  return now;
}
```

## 2.2 DSU

```
int parent[100005]

  void
  init()
{
  for (int i = 0; i < 100005; i++)
    parent[i] = i;
}

int find_root(int x)
{
  if (x == parent[x])
    return x;
  return parent[x] = find_root(parent[x]);
}

bool Same(int a, int b)
{
  return find_root(a) == find_root(b);
}

void Union(int a, int b)
{
  // 將a併進b
  parent[find_root(a)] = find_root(b);
}
```

# 3 BigNum

## 3.1 add

```
#include <stdio.h>
#include <string.h>
```

```c
#include <stdlib.h>

void swap(char *a, char *b)
{
  char temp = *a;
  *a = *b;
  *b = temp;
}

char *add(char *s1, char *s2)
{
  char *ans = (char *)malloc(sizeof(char) * 200);
  int len1 = strlen(s1);
  int len2 = strlen(s2);
  for (int i = len1; i < 200; i++)
  {
    s1[i] = '0';
  }
  for (int i = len2; i < 200; i++)
  {
    s2[i] = '0';
  }
  for (int i = 0; i < len1 / 2; i++)
  {
    swap(&s1[i], &s1[len1 - i - 1]);
  }
  for (int i = 0; i < len2 / 2; i++)
  {
    swap(&s2[i], &s2[len2 - i - 1]);
  }

  int carry = 0, len3 = 0;
  for (int i = 0; i < 200; i++)
  {
    int num1 = s1[i] - '0';
    int num2 = s2[i] - '0';
    ans[i] = (
        char)(((num1 + num2 + carry) % 10) + (int)'0');
    if (ans[i] != '0')
      len3 = i + 1;
    carry = (num1 + num2 + carry) / 10;
  }
  for (int i = 0; i < len3 / 2; i++)
  {
    swap(&ans[i], &ans[len3 - i - 1]);
  }
  for (int i = len3; i < 200; i++)
  {
    ans[i] = '\0';
  }
  return ans;
}

int main()
{
  char *s1 = (char *)malloc(sizeof(char) * 200);
  char *s2 = (char *)malloc(sizeof(char) * 200);
  scanf("%s %s", s1, s2);
  printf("%s", add(s1, s2));
  return 0;
}
```

## 3.2 pow

```c
// 大數乘法
#include <stdio.h>
#include <math.h>
#include <string.h>
#define M 10005
char s1[M], s2[M], s[M];
int a[M], b[M], c[M];
int main()
{
  int i, j, m, n, k;
  while (~scanf("%s%s"

                ,
                s1, s2))
  {
    memset(c, 0, sizeof(c));
    n = strlen(s1);
    m = strlen(s2);
    k = n + m; // 保證相乘後的位數不會大於k
    printf("s1的長度=%d s2的長度=%d\n", n, m);
    /*把字串s1和s2逆序用數字排列*/
    for (i = 0; i < n; i++)
      a[i] = s1[n - i - 1] - '0';
    for (i = 0; i < m; i++)
      b[i] = s2[m - 1 - i] - '0';
    /* 乘運算*/

    for (i = 0; i < n; i++)
      for (j = 0; j < m; j++)
        c[i + j] += a[i] * b[j];
    for (i = 0; i <= k; i++)
    {
      if (c[i] >= 10)
      {
        c[i + 1] += c[i] / 10;
        c[i] %= 10;
      }
    }
    /*去除前導0*/
    i = k;
    while (c[i] ==

           0)
      i--;
    /*判斷兩個非負數之積是否為0，以及逆序列印c[]*/
    if (i < 0)
      printf("0");
    else
    {
      for (; i >= 0; i--)
        printf("%d", c[i]);
    }
    printf("\n");
  }
  return 0;
}
```

## 4 Math

### 4.1 gcd

```cpp
int gcd(int a, int b)
{
  return b == 0 ? a : gcd(b, a % b);
}
int lcm(int a, int b)
{
  return a * b / gcd(a, b);
}
pair<int, int> ext_gcd
    (int a, int b) //擴展歐幾里德 ax+by = gcd(a,b)
{
  if (b == 0)
    return {1, 0};
  if (a == 0)
    return {0, 1};
  int x, y;
  tie(x, y) = ext_gcd(b % a, a);
  return make_pair(y - b * x / a, x);
}
```

### 4.2 快速冪

```cpp
//x^y % p
ll func(ll x,ll y,ll p){
  ll res = 1;
  while(y != 0){
    if(y%2==1){
      res *= x;
      res %=p;
    }
    x *= x;
    y /= 2;// 5^8 => (5^2)^4
    x %= p;//((5^2) % 7)^4
  }
  return res;
}
```