Cracking the Coding Interview – Solutions

Heyan Huang

Contents

Chapter 1

Arrays and Strings

1.1 Implement an algorithm to determine if a string has all unique characters. What if you can not use additional data structures?

```
#include <iostream>
#include <cstring>
using namespace std;

bool isUnique1(string s) {
  bool set[256];
  memset(set, 0, sizeof(set));
  int len = s.length();
  for (int i = 0; i < len; i++) {
    int val = (int)s[i];
    if (set[val])
      return false;
    set[val] = true;
  }
  return true;
}

bool isUnique2(string s) {
  int A[8];
  memset(A, 0, sizeof(A));
  int len = s.length();
  for (int i = 0; i < len; i++) {
    int val = (int)s[i];
    int index = val/32;
    int shift = val % 32;
    if (A[index] & (1<<shift))
      return false;
    // A[index] ^ (1<<shift); // or A[index] | (1<<shift);  They are wrong!!!
    A[index] ^= (1<<shift); // or A[index] |= (1<<shift);
  }
  return true;
}

bool isUnique3(string s) {
    int check = 0;
    int len = s.length();
```

```
    for(int i=0; i < len; ++i) {
        int v = (int)(s[i]-'a');
        if(check & (1 << v)) return false;
        check |= (1 << v);
    }
    return true;
}

int main() {
    string s1 = "i am hawstein.";
    string s2 = "abcdefghijklmnopqrstuvwxyzABCD1234567890";
    cout << isUnique1(s1) << " " << isUnique1(s2) << endl;
    cout << isUnique2(s1) << " " << isUnique2(s2) << endl;
    return 0;
}
```

```
def has_unique_chars(string):
    # Strategy: Linearly check if the char of the string belongs to the dictionary, other
    # Time complexity is O(n) and space complexity is O(n)
    dictionary = {} #technically, a boolean array (for the char set) would do the job for
    for i in string:
        if i in dictionary:
            return False
        else:
            dictionary[i] = True
    return True

# What if you can't use an additional data structure?

def has_unique_chars2(string):
    # Stragety: Check every char of the string with every other char of the string
    # Time complexity is O(n^2) and space complexity is O(1)
    for i in xrange(len(string)):
        for j in xrange(i + 1, len(string)):
            if string[i] == string[j]:
                return False
    return True

# If destroying the input is allowed:

def has_unique_chars3(string):
    # Strategy: Sort the string inline and linearly check the string for identical neighb
    # Time complexity is O(n * log(n) + n => n * log(n)) and space complexity is O(1)
    string = ''.join(sorted(string)) #technically, this creates a new list and then joins
    for i in xrange(len(string) - 1):
        if string[i] == string[i+1]:
            return False
    return True

if __name__ == '__main__':
    case1 = 'abcdefghijklmnopqrstuvwxyz'
    case2 = 'abcdefghijklmabcdefghijklm'
    print has_unique_chars(case1)
    print has_unique_chars(case2)
    print has_unique_chars2(case1)
```

```
40      print has_unique_chars2(case2)
41      print has_unique_chars3(case1)
42      print has_unique_chars3(case2)
```

1.2 Write code to reverse a C-Style String. (C-String means that abcd is represented as five characters, including the null character.)

```
1  #include <iostream>
2  #include <cstring>
3  using namespace std;
4
5  void swap(char &a, char &b){   // not understanding yet
6      a = a^b;
7      b = a^b;
8      a = a^b;
9  }
10
11 void reverse(char* s) {
12   int len = strlen(s);
13   char tmp;
14   for (int i = 0; i < len/2; ++i) {
15     tmp = s[len-1-i];
16     s[len-1-i] = s[i];
17     s[i] = tmp;
18   }
19 }
20
21 void reverseP(char* s) {
22   if(!s) return; // I forgot this line at beginning
23   char *f = s, *b = s;
24   char tmp;
25   while(*b) b++;
26   b--;
27   //while(f != b) {
28   while(f < b) {
29     tmp = *f;
30     *f = *b;
31     *b = tmp;
32     f++;
33     b--;
34   }
35 }
36
37 void reverseFinal(char* str) {
38   char * end = str;
39   char tmp;
40   if (str) {           // equvilent to: if(!str) return;
41     while(*end) ++end;
42     --end;
43     while(str < end) {
44       tmp = *str;
45       *str++ = *end;
46       *end-- = tmp;
47     }
48   }
```

```
49 }
50
51 void reverse2(char *s){
52     int n = strlen(s);
53     for(int i=0; i<n/2; ++i)
54         swap(s[i], s[n-i-1]);
55 }
56
57 void reverse1(char *s){
58     if(!s) return;
59     char *p = s, *q = s;
60     while(*q) ++q;
61     --q;
62     while(p < q)
63         swap(*p++, *q--);
64 }
65
66 int main(){
67   char s[] = "1234567890";
68   char a[] = "abcdefg";
69   char b[]= "hijklmn";
70   char c[]= "hijklmn";
71   char d[]= "hijklmn";
72   reverse(b);
73   reverseP(c);
74   reverseFinal(d);
75   reverse1(s);
76   reverse2(a);
77   cout << "reverse:" << b << endl;
78   cout << "reverseP:" << c << endl;
79   cout << "reverseFinal:" << d << endl;
80   cout << "reverse1:" << s << endl;
81   cout << "reverse2:" << a << endl;
82   return 0;
83 }
```

1.3 Design an algorithm and write code to remove the duplicate characters in a string without using any additional buffer. NOTE: One or two additional variables are fine. An extra copy of the array is not. FOLLOW UP Write the test cases for this method.

```
#include<iostream>
#include<cstring>
using namespace std;

string removeDuplicate1(char s[]) {
    int len = strlen(s);
    if (len < 2)
        return s;
    string str = "";

    int a = 0;
    for (int i = 0; i < len; ++i) {
        int shift = (int)(s[i] - 'a');
        if ( (a & (1 << shift)) == 0 ) {
            str += s[i];
            a |= (1 << shift);
        }
    }
    return str;
}

string removeDuplicate2(char s[]) {
    int len = strlen(s);
    if (len < 2)
        return s;
    string str = "";

    for(int i = 0; i < len; ++i) {
        if (s[i] != '\0') {
            str += s[i];
            for (int j = i+1; j < len; ++j)
                if ( s[j] == s[i] )
                    s[j] = '\0';
        }
    }
    return str;
}

void removeDuplicate3(char s[]) {
    int len = strlen(s);
    if (len < 2) return;

    int p = 0;
    for(int i = 0; i < len; ++i) {
        if (s[i] != '\0') {
            s[p++] = s[i];
            for (int j = i+1; j < len; ++j)
                if ( s[j] == s[i] )
                    s[j] = '\0';
        }
    }
```

```
52      s[p] = '\0';
53  }
54
55  void removeDuplicate4(char s[]) {
56      int len = strlen(s);
57      if (len < 2) return;
58
59      bool flag[256];
60      memset(flag, 0, sizeof(flag));
61
62      int p = 0;
63      for (int i = 0; i < len; ++i) {
64          if (!flag[s[i]]) {
65              s[p++] = s[i];
66              flag[s[i]] = true;
67          }
68      }
69      s[p] = '\0';
70  }
71
72  void removeDuplicate5(char s[]) {
73      int len = strlen(s);
74      if (len < 2) return;
75
76      int a = 0, p = 0;
77      for (int i = 0; i < len; ++i) {
78          int shift = (int)(s[i] - 'a');
79          if ( !(a & (1 << shift)) ) {
80              s[p++] = s[i];
81              a |= 1 << shift;
82          }
83      }
84      s[p] = '\0';
85  }
86
87  int main() {
88      char a[] = "abcdefg";
89      char b[] = "aaaaabbbbbccccc";
90      char c[] = "";
91      char d[] = "abcdefgadchijkadefgabc";
92      char e[] = "aaaaaaaaa";
93      char a2[] = "abcdefg";
94      char b2[] = "aaaaabbbbbccccc";
95      char c2[] = "";
96      char d2[] = "abcdefgadchijkadefgabc";
97      char e2[] = "aaaaaaaaa";
98      cout << removeDuplicate1(a) << "␣" << removeDuplicate2(a2) << endl;
99      cout << removeDuplicate1(b) << "␣" << removeDuplicate2(b2) << endl;
100     cout << removeDuplicate1(c) << "␣" << removeDuplicate2(c2) << endl;
101     cout << removeDuplicate1(d) << "␣" << removeDuplicate2(d2) << endl;
```

1.4 Write a method to decide if two strings are anagrams or not.

```
#include<iostream>
#include<cstring>
#include<algorithm>
using namespace std;

bool isAnagram1 (string s, string t) {
    if (s == "" || t == "") return false;
    if (s.length() != t.length()) return false;

    sort(&s[0], &s[0]+s.length());
    sort(&t[0], &t[0]+t.length());
    if (s == t) return true;
    else return false;
}

bool isAnagram2 (string s, string t) {
    if (s == "" || t == "") return false;
    if (s.length() != t.length()) return false;

    int a[256];
    memset(a, 0, sizeof(a));

    int len = s.length();
    for (int i = 0; i < len; ++i) {
        ++a[s[i]];
        --a[t[i]];
    }

    for (int i = 0; i < 256; ++i)
        if (a[i])
            return false;
    return true;
}

int main() {
    string s1 = "aeiou";
    string s2 = "uoeai";
    cout << isAnagram1(s1, s2) << endl;
    cout << isAnagram2(s1, s2) << endl;
    return 0;
}
```

1.5 Write a method to replace all spaces in a string with %20.

```
#include <iostream>
#include <cstring>
#include <cstdlib>
using namespace std;

char* replace1 (char* c) {
    if (c == NULL) return NULL; // didn't notice this one

    int len = strlen(c);    // didn't notice this one
    if (len == 0) return NULL;

    int cnt = 0;
    for (int i = 0; i < len; ++i) {
        if (c[i] == ' ')
            ++cnt;
    }

    char* cc = new char[len + 2*cnt];    // something I missed here!!!
    int p = 0;
    for (int i = 0; i < len; ++i) {
        if (c[i] == ' ') {
            cc[p++] = '%';
            cc[p++] = '2';
            cc[p++] = '0';
        } else
            cc[p++] = c[i];
    }
    cc[p] = '\0';
    return cc;
}

void replace2 (char* c) {
    if (c == NULL) return; // didn't notice this one

    int len = strlen(c);    // didn't notice this one
    if (len == 0) return;

    int cnt = 0;
    for (int i = 0; i < len; ++i) {
        if (c[i] == ' ')
            ++cnt;
    }

    int p = len + 2 * cnt;
    c[p--] = '\0';     // the space must be allocated first!
    for (int i = len-1; i >= 0; --i) {
        if (c[i] == ' ') {
            c[p] = '0';
            c[p - 1] = '2';
            c[p - 2] = '%';
            p -= 3;
        } else {
            c[p] = c[i];
```

```
54                 --p;
55             }
56         }
57     }
58
59     int main() {
60         const int len = 100;
61         char s[len] = "I am beautiful!";
62         cout << replace1(s) << endl;
63         //replace2(s);
64         //cout << s << endl;
65         return 0;
66     }
```

1.6 Given an image represented by an NxN matrix, where each pixel in the image is 4 bytes, write a method to rotate the image by 90 degrees. Can you do this in place?

```
1  #include <iostream>
2  using namespace std;
3
4  void swap(int &a, int &b) {
5      int tmp = a;
6      a = b;
7      b = tmp;
8  }
9
10 void transpose(int a[][4], int n) {
11     for (int i = 0; i < n; ++i )
12         for (int j = i+1; j < n; ++j )
13             swap(a[i][j], a[j][i]);
14     for (int i = 0; i < n/2; ++i )
15         for (int j = 0; j < n; ++j )
16             swap(a[i][j], a[n-1-i][j]);
17 }
18
19 int main() {
20     int a[4][4] = {
21         {1, 2, 3, 4},
22         {5, 6, 7, 8},
23         {9, 10, 11, 12},
24         {13, 14, 15, 16}
25     };
26     for (int i = 0; i < 4; ++i ) {
27         for (int j = 0; j < 4; ++j ) {
28             cout << a[i][j] << '\t';
29         }
30         cout << endl;
31     }
32     cout << endl;
33     transpose(a, 4);
34     for (int i = 0; i < 4; ++i ) {
35         for (int j = 0; j < 4; ++j ) {
36             cout << a[i][j] << '\t';
37         }
38         cout << endl;
```

```
39      }
40      return 0;
41  }
```

1.7 Write an algorithm such that if an element in an MxN matrix is 0, its entire row and column is set to 0.

```
1   #include <iostream>
2   #include <cstring>
3   #include <cstdio>
4   using namespace std;
5
6   void zero(int **a, int m, int n) {
7       bool row[m], col[n];
8       memset(row, 0, sizeof(row));    // seems to have problem LeetCode reference
9       memset(col, 0, sizeof(col));
10      for (int i = 0; i < m; ++i )
11          for (int j = 0; j < n; ++j )
12              if (a[i][j] == 0) {
13                  row[i] = true;
14                  col[j] = true;
15              }
16      for (int i = 0; i < m; ++i ) {
17          for (int j = 0; j < n; ++j ) {
18              if (row[i] || col[j])
19                  a[i][j] = 0;
20          }
21      }
22  }
23
24  int main() {
25      freopen("sev.txt", "r", stdin);
26
27      int m, n;
28      cin >> m >> n;
29      int **a;
30      for (int i = 0; i < m; ++i )
31          a[i] = new int[n];
32      for (int i = 0; i < m; ++i ){
33          for (int j = 0; j < n; ++j ) {
34              cin >> a[i][j];
35          }
36      }
37
38      for (int i = 0; i < m; ++i ) {
39          for (int j = 0; j < n; ++j ) {
40              cout << a[i][j] << '\t';
41          }
42          cout << endl;
43      }
44      cout << endl;
45
46      zero(a, m, n);
47      for (int i = 0; i < m; ++i ) {
48          for (int j = 0; j < n; ++j ) {
49              cout << a[i][j] << '\t';
```

```
50          }
51          cout << endl;
52      }
53
54      fclose(stdin);
55      return 0;
56  }
```

1.8 Assume you have a method isSubstring which checks if one word is a substring of another. Given two strings, s1 and s2, write code to check if s2 is a rotation of s1 using only one call to isSubstring (i.e., waterbottle is a rotation of erbottlewat).

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  bool isSubstring(string s1, string s2) {
6      if ( s1.find(s2) != string::npos)  // the part that I am not that familiar!!!
7          return true;
8      else return false;
9  }
10
11 bool isRotation(string s1, string s2) {
12     if ( (s1.length() != s2.length()) || s1.length() <= 0 )
13         return false;
14     return isSubstring(s1+s1, s2);
15 }
16
17 int main() {
18     string s1 = "waterbottle";
19     string s2 = "erbottlewat";
20     cout << isRotation(s1, s2) << endl;
21     return 0;
22 }
```

Chapter 2

Linked Lists

2.1 Write code to remove duplicates from an unsorted linked list.

FOLLOW UP

How would you solve this problem if a temporary buffer is not allowed?

.

Solution:

C++ (java)exthash_mapC++ (LinuxWindows) int

O(n):

```
1  void removedulicate(node *head){
2      if(head==NULL) return;
3      node *p=head, *q=head->next;
4      hash[head->data] = true;
5      while(q){
6          if(hash[q->data]){
7              node *t = q;
8              p->next = q->next;
9              q = p->next;
10             delete t;
11         }
12         else{
13             hash[q->data] = true;
14             p = q; q = q->next;
15         }
16     }
17 }
```

() O(n2)

```
1  void removedulicate1(node *head){
2      if(head==NULL) return;
3      node *p, *q, *c=head;
4      while(c){
5          p=c; q=c->next;
6          int d = c->data;
7          while(q){
8              if(q->data==d){
9                  node *t = q;
10                 p->next = q->next;
11                 q = p->next;
12                 delete t;
13             }
```

```
14              else{
15                  p = q; q = q->next;
16              }
17          }
18          c = c->next;
19      }
20  }
```

```
1   #include <iostream>
2   #include <cstring>
3   using namespace std;
4
5   typedef struct node {
6       int data;
7       node* next;
8   } node;
9
10  bool hash[100];
11
12  node* init(int a[], int n) {
13      node *head, *p;
14      for (int i = 0; i < n; ++i) {
15          node * nd = new node();
16          nd->data = a[i];
17          if (i == 0) {
18              head = p = nd; // set head
19              continue;
20          }
21          p->next = nd;
22          //p = nd;           // same as followed line
23          p = p->next;
24      }
25      return head;
26  }
27
28  void removeDuplicate1(node* head) {
29      if (head == NULL) return;
30
31      node* p = head, *q = head->next;
32      hash[head->data] = true;
33
34      while(q) {
35          if (hash[q->data]) {
36              node* tmp = q;
37              q = q->next;
38              p->next = q;
39              delete tmp;
40          } else {
41              hash[q->data] = true;
42              p = q;
43              q = q->next;
44          }
45      }
46  }
47
48  void removeDuplicate2(node* head) {
49      if (head == NULL) return;
50
```

```
51      node* p = head, *curr = head->next, *prev;
52      while(p) {
53          prev = p;
54          curr = p->next;
55          while(curr) {
56              if (curr->data == p->data) {
57                  node* tmp = curr;
58                  prev->next = curr->next;
59                  curr = curr->next;  // same as: curr = prev->next;
60                  delete tmp;
61              } else {
62                  prev = curr;
63                  curr = curr->next;
64              }
65          }
66          p = p->next;
67      }
68  }
69
70  void print(node* head) {
71      node * curr = head;
72      while (curr) {
73          cout << curr->data << ' ';
74          curr = curr->next;
75      }
76      cout << endl;
77  }
78
79  int main() {
80      int n = 10;
81      int a[] = {3, 2, 1, 3, 5, 6, 2, 6, 3, 1};
82      memset(hash, false, sizeof(hash));
83      node *head = init(a, 10);
84      print(head);
85      //removeDuplicate1(head);
86      removeDuplicate2(head);
87      print(head);
88      return 0;
89  }
```

2.2 Implement an algorithm to find the nth to last element of a singly linked list.

n

Solution:

n n n

n

n11

```
1  node *pp;
2  int nn;
3  void findNthToLast1(node *head){
4      if(head==NULL) return;
5      findNthToLast1(head->next);
6      if(nn==1) pp = head;
7      --nn;
8  }
```

( ) n ntricky

```
node* findNthToLast(node *head, int n){
    if(head==NULL || n < 1) return NULL;
    node *p=head, *q=head;
    while(n>0 && q){
        q = q->next;
        --n;
    }
    if(n > 0) return NULL;
    while(q){
        p = p->next;
        q = q->next;
    }
    return p;
}
```

:P

```
#include <iostream>
using namespace std;

typedef struct node {
    int data;
    node* next;
} node;

node* init(int a[], int n) {
    node *head, *p;
    for (int i = 0; i < n; ++i) {
        node * nd = new node();
        nd->data = a[i];
        if (i == 0) {
            head = p = nd; // set head
            continue;
        }
        p->next = nd;
        p = p->next;
    }
    return head;
}

// traverse the list twice, my method
int findNthToLast1(node* head, int n) {
    if (head == NULL) return 0;
    int m, cnt = 1;
    node *prev = head, *curr = prev->next;
    while(curr) {
        prev = curr;
        curr = curr->next;
        ++cnt;
    }
    m = cnt;
    cnt = 0;
    prev = head;
    curr = prev->next;
    if (curr==NULL && n == 1)
        return head->data;

    while(curr) {
        if (cnt == m - n)
```

```
43              return prev->data;
44          else {
45              prev = curr;
46              curr = curr->next;
47              ++cnt;
48          }
49      }
50  }
51
52  node* pp;
53  int nn;
54
55  void findNthToLast2(node* head) {
56      if (head == NULL) return;
57      findNthToLast2(head->next);
58      if (nn==1)
59          pp = head;
60      --nn;
61  }
62
63  // two pointer with n nodes separated
64  node* findNthToLast3(node* head, int n) {
65      if ( (head == NULL) || n < 1) return NULL; // forgot to consider
66
67      node * prev = head, *curr = head;
68      while (n > 0 && curr) {
69          curr = curr->next;
70          --n;
71      }
72      if (n > 0) return NULL;  // forgot to consider
73
74      while(curr) {
75          prev = prev->next;
76          curr = curr->next;
77      }
78      return prev;
79  }
80
81  void print(node* head) {
82      node * curr = head;
83      while (curr) {
84          cout << curr->data << ' ';
85          curr = curr->next;
86      }
87      cout << endl;
88  }
89
90  int main() {
91      int n = 10;
92      int a[] = {3, 2, 1, 3, 5, 6, 2, 6, 3, 1};
93      node *head = init(a, 10);
94      print(head);
95
96      cout << findNthToLast1(head, 4) << endl;
97
98      node *p = findNthToLast3(head, 4);
99      if (p) cout << p->data << endl;
100     else cout << "the length of link is not long enough" << endl;
101
```

```
102     nn = 4;
103     findNthToLast2(head);
104     if (pp) cout << pp->data << endl;
105
106     return 0;
107 }
```

2.3 Implement an algorithm to delete a node in the middle of a single linked list, given only access to that node.
EXAMPLE
Input: the node c from the linked list a->b->c->d->e
Result: nothing is returned, but the new linked list looks like a->b->d->e

a->b->c->d->ec
a->b->d->e
Solution:
c cdc nextedelete dokd a->b->c->edc a->b->d->eOK
()c1.2. 3.4.12dccnextd nextdOK43c cokc cc(b)next 0
CTCI c

```
1  bool remove(node *c){
2      if(c==NULL || c->next==NULL) return false;
3      node *q = c->next;
4      c->data = q->data;
5      c->next = q->next;
6      delete q;
7      return true;
8  }
```

```
1  #include <iostream>
2  using namespace std;
3
4  typedef struct node{
5      int data;
6      node *next;
7  }node;
8
9  node* init(int a[], int n){
10     node *head, *p;
11     for(int i=0; i<n; ++i){
12         node *nd = new node();
13         nd->data = a[i];
14         if(i==0){
15             head = p = nd;
16             continue;
17         }
18         p->next = nd;
19         p = p->next;
20     }
21     return head;
22 }
23
24 bool remove(node *c){
25     if (c==NULL || c->next == NULL) return false;
26     //if (c->next == NULL) { //c0
27     //     delete c;
28     //     return;
29     //}
```

```
30      node* tmp = c->next;
31      c->data = tmp->data;
32      c->next = tmp->next;
33      delete tmp;
34      return true;
35  }
36
37  void print(node *head){
38      while(head){
39          cout<<head->data<<" ";
40          head = head->next;
41      }
42      cout<<endl;
43  }
44
45  int main(){
46      int n = 10;
47      int a[] = {9, 2, 1, 3, 5, 6, 2, 6, 3, 1};
48      node *head = init(a, n);
49      int cc = 3;
50      node *c = head;
51      for(int i=1; i<cc; ++i) c = c->next;
52      print(head);
53      if(remove(c))
54          print(head);
55      else
56          cout<<"failure"<<endl;
57      return 0;
58  }
```

2.4 You have two numbers represented by a linked list, where each node contains a single digit. The digits are stored in reverse order, such that the 1s digit is at the head of the list. Write a function that adds the two numbers and returns the sum as a linked list.

EXAMPLE

Input: (3 -> 1 -> 5) + (5 -> 9 -> 2)

Output: 8 -> 0 -> 8

(3 -> 1 -> 5), (5 -> 9 -> 2)

8 -> 0 -> 8

Solution: 1.2.3.

```
1   #include <iostream>
2   #include <cmath>
3   using namespace std;
4
5   typedef struct node{
6       int data;
7       node *next;
8   }node;
9
10  node* init(int a[], int n){
11      node *head, *p;
12      for(int i=0; i<n; ++i){
13          node *nd = new node();
14          nd->data = a[i];
15          if(i==0){
16              head = p = nd;
17              continue;
18          }
```

```
19          p->next = nd;
20          p = p->next;
21      }
22      return head;
23  }
24
25  // single list, return single list
26  node* sumAdd(node* head) {
27      if (head == NULL || head->next == NULL) return NULL;
28
29      node *prev = head, *curr = head->next;
30      int a = prev->data, cnt = 0;
31      while(curr->next) {
32          prev = prev->next;
33          curr = curr->next->next;
34          ++cnt;
35          a += prev->data * pow(10, cnt);
36      }
37
38      curr = prev->next;
39      int b = curr->data;
40      cnt = 0;
41      while(curr->next) {
42          curr = curr->next;
43          ++cnt;
44          b += curr->data * pow(10, cnt);
45      }
46
47      int result = a + b;
48      if (result >= 0) {
49          node* newhead = new node();
50          newhead->data = result % 10;
51          newhead->next = NULL;
52
53          result = result / 10;
54          node * p = newhead;
55
56          while (result > 0) {
57              node* curr = new node();
58              curr->data = result % 10;
59              curr->next = NULL;
60              p->next = curr;
61              p = p->next;
62              result = result / 10;
63          }
64          return newhead;
65      } else
66          return NULL;
67  }
68
69  // double lists, return a list
70  node* addlink(node *p, node *q){
71      if (p == NULL) return q;
72      if (q == NULL) return p;
73
74      int result = 0;
75      node *curr = NULL, *newhead;
76      while (p && q) {
77          result += p->data + q->data;
```

```
78          node* tmp = new node();
79          tmp->data = result % 10;
80          tmp->next = NULL;
81          if (curr) {
82              curr->next = tmp;
83              curr = curr->next;
84          } else
85              newhead = curr = tmp;
86          p = p->next;
87          q = q->next;
88          result = result / 10;
89      }
90
91      node* r;
92      if (p) r = p;
93      else r = q;
94      while (r) {
95          result += r->data;
96          node* tmp = new node();
97          tmp->data = result % 10;
98          tmp->next = NULL;
99          curr->next = tmp;
100         curr = curr->next;
101         r = r->next;
102         result = result / 10;
103     }
104
105     if (result > 0) { //
106         node* tmp = new node();
107         tmp->data = result;
108         tmp->next = NULL;
109         curr->next = tmp;
110     }
111     return newhead;
112 }
113
114 void print(node *head){
115     while(head){
116         cout<<head->data<<" ";
117         head = head->next;
118     }
119     cout<<endl;
120 }
121
122 int main(){
123     int n = 6;
124     int a[] = {3, 1, 4, 5, 9, 2};
125     node *head = init(a, n);
126     node* curr = sumAdd(head);
127     if (curr) print(curr);
128     else cout << "fail!!!" << endl;
129     cout << endl;
130
131     int n2 = 4;
132     int a2[] = {1, 2, 9, 3};
133     int m = 3;
134     int b2[] = {9, 9, 2};
135
136     node *p = init(a2, n2);
```

```
137     node *q = init(b2, m);
138     node *res = addlink(p, q);
139     if(p) print(p);
140     if(q) print(q);
141     if(res) print(res);
142
143     return 0;
144 }
```

2.5 Given a circular linked list, implement an algorithm which returns node at the beginning of the loop.

DEFINITION

Circular linked list: A (corrupt) linked list in which a nodes next pointer points to an earlier node, so as to make a loop in the linked list.

EXAMPLE

input: A -> B -> C -> D -> E -> C [the same C as earlier]

output: C

A -> B -> C -> D -> E -> C [C]

C

Solution:

http://hawstein.com/posts/2.5.html

tricky (21)

(D)kn(nDK8) fastslowheadk m2m n

2m - m = pn –> m = pn, p

m+1=pn+1(m) 1pn+1-k(k)pn+1-k q qn-(pn+1-k)+1=(q-p)n+k(D) (D)

(q-p)n+k <= n

qpnkq-p <= 0 q-p <= 0k

(q-p)n+k <= k

(q-p)n+khead k

(q-p)n+k (q-p)n+k

k - [(q-p)n + k] = (p-q)n ()

q-p <= 0p-q >= 0 (1)(p-q)

```
1  node* loopstart(node *head){
2      if(head==NULL) return NULL;
3      node *fast = head, *slow = head;
4      while(fast && fast->next){
5          fast = fast->next->next;
6          slow = slow->next;
7          if(fast==slow) break;
8      }
9      if(!fast || !fast->next) return NULL;
10     slow = head;
11     while(fast!=slow){
12         fast = fast->next;
13         slow = slow->next;
14     }
15     return fast;
16 }
```

tricky :p

true true OK

C++mapO(1) mapRB treeRB tree

```
map<node*, bool> hash;
node* loopstart1(node *head){
    while(head){
        if(hash[head]) return head;
        else{
            hash[head] = true;
            head = head->next;
        }
    }
    return head;
}
```

Chapter 3

Stacks and Queues

3.1 Describe how you could use a single array to implement three stacks.

Solution:
1 1 -1
33 3

```
1  class stack3{
2  private:
3      int *buf;
4      int ptop[3];
5      int size;
6  public:
7      stack3(int size = 300){
8          buf = new int[size*3];
9          ptop[0]=ptop[1]=ptop[2]=-1;
10         this->size = size;
11     }
12     ~stack3(){
13         delete[] buf;
14     }
15
16     void push(int stackNum, int val){
17         int idx = stackNum*size + ptop[stackNum] + 1;
18         buf[idx] = val;
19         ++ptop[stackNum];
20     }
21     void pop(int stackNum){
22         --ptop[stackNum];
23     }
24     int top(int stackNum){
25         int idx = stackNum*size + ptop[stackNum];
26         return buf[idx];
27     }
28     bool empty(int stackNum){
29         return ptop[stackNum]==-1;
30     }
31 };
```

```
typedef struct node{
    int val;
    int preIdx;
} node;
```

```
class stack3_1{
private:
    node *buf;
    int ptop[3];
    int totalSize;
    int cur;
public:
    stack3_1(int totalSize = 900){
        buf = new node[totalSize];
        ptop[0]=ptop[1]=ptop[2]=-1;
        this->totalSize = totalSize;
        cur = 0;
    }
    ~stack3_1(){
        delete[] buf;
    }

    void push(int stackNum, int val){
        buf[cur].val = val;
        buf[cur].preIdx = ptop[stackNum];
        ptop[stackNum] = cur;
        ++cur;
    }
    void pop(int stackNum){
        ptop[stackNum] = buf[ptop[stackNum]].preIdx;
    }
    int top(int stackNum){
        return buf[ptop[stackNum]].val;
    }
    bool empty(int stackNum){
        return ptop[stackNum]==-1;
    }
};
```

cur pop cur

cur popcur curcurpush curcur

```
#include <iostream>
using namespace std;

class stack3 {
private:
    int *buf;
    int ptop[3];
    int size;
public:
    stack3(int size = 300) {
        buf = new int[size*3];
        ptop[0] = ptop[1] = ptop[2] = -1;
        this->size = size;
    }
```

```
15      ~stack3() {
16          delete[] buf;
17      }
18
19      void push(int stackNum, int val) {
20          int idx = stackNum*size + ptop[stackNum] + 1;
21          buf[idx] = val;
22          ++ptop[stackNum];
23      }
24      void pop(int stackNum) {
25          --ptop[stackNum];
26      }
27      int top(int stackNum) {
28          int idx = stackNum*size + ptop[stackNum];
29          return buf[idx];
30      }
31      bool empty(int stackNum) {
32          return ptop[stackNum] == -1;
33      }
34  };
35
36  typedef struct node {
37      int val;
38      int preIdx;
39  } node;
40
41  class stack3_1{
42  private:
43      node* buf;
44      int ptop[3];
45      int totalSize;
46      int curr;
47  public:
48      stack3_1(int totalSize = 900) {
49          buf = new node[totalSize];
50          ptop[0] = ptop[1] = ptop[2] = -1;
51          this->totalSize = totalSize;
52          curr = 0;
53      }
54      ~stack3_1() {
55          delete[] buf;
56      }
57
58      void push(int stackNum, int val) {
59          buf[curr].val = val;
60          buf[curr].preIdx = ptop[stackNum];
61          ptop[stackNum] = curr;
62          ++curr;
63      }
64      void pop(int stackNum) {
65          ptop[stackNum] = buf[ptop[stackNum]].preIdx;
66      }
67      int top(int stackNum) {
68          return buf[ptop[stackNum]].val;
69      }
70      bool empty(int stackNum) {
71          return ptop[stackNum] == -1;
72      }
73  };
```

```
74
75  int main(){
76
77      //stack3 mystack;
78      stack3_1 mystack;
79      for(int i=0; i<10; ++i)
80          mystack.push(0, i);
81      for(int i=10; i<20; ++i)
82          mystack.push(1, i);
83      for(int i=100; i<110; ++i)
84          mystack.push(2, i);
85      for(int i=0; i<3; ++i)
86          cout<<mystack.top(i)<<" ";
87      cout<<endl;
88
89      for(int i=0; i<3; ++i){
90          mystack.pop(i);
91          cout<<mystack.top(i)<<" ";
92      }
93      cout<<endl;
94
95      mystack.push(0, 111);
96      mystack.push(1, 222);
97      mystack.push(2, 333);
98      for(int i=0; i<3; ++i)
99          cout<<mystack.top(i)<<" ";
100     cout<<endl;
101
102     return 0;
103 }
```

3.2 How would you design a stack which, in addition to push and pop, also has a function min which returns the minimum element? Push, pop and min should all operate in O(1) time.

pushpopmin pushpopminO(1)

Solution:

push pop

```
1  const int MAX_INT = ~(1<<31);//2147483647
2
3  typedef struct node{
4      int val, min;
5  }node;
6
7  class StackWithMin{
8  public:
9      StackWithMin(int size=1000){
10         buf = new node[size];
11         buf[0].min = MAX_INT;
12         cur = 0;
13     }
14     ~StackWithMin(){
15         delete[] buf;
16     }
17     void push(int val){
18         buf[++cur].val = val;
19         if(val<buf[cur-1].min) buf[cur].min = val;
20         else buf[cur].min = buf[cur-1].min;
21     }
```

```
22      void pop(){
23          --cur;
24      }
25      int top(){
26          return buf[cur].val;
27      }
28      bool empty(){
29          return cur==0;
30      }
31      int min(){
32          return buf[cur].min;
33      }
34
35  private:
36      node *buf;
37      int cur;
38  };
```

110000 1110000 1 110000 s1s2 s1s2s2 s2s2 s2s1pop pops1s2 s2s1

```
1   class stack{
2   public:
3       stack(int size=1000){
4           buf = new int[size];
5           cur = -1;
6       }
7       ~stack(){
8           delete[] buf;
9       }
10      void push(int val){
11          buf[++cur] = val;
12      }
13      void pop(){
14          --cur;
15      }
16      int top(){
17          return buf[cur];
18      }
19      bool empty(){
20          return cur==-1;
21      }
22
23  private:
24      int *buf;
25      int cur;
26  };
27
28  class StackWithMin1{
29  public:
30      StackWithMin1(){
31
32      }
33      ~StackWithMin1(){
34
35      }
36      void push(int val){
37          s1.push(val);
38          if(val<=min())
39              s2.push(val);
```

```
40      }
41      void pop(){
42          if(s1.top()==min())
43              s2.pop();
44          s1.pop();
45      }
46      int top(){
47          return s1.top();
48      }
49      bool empty(){
50          return s1.empty();
51      }
52      int min(){
53          if(s2.empty()) return MAX_INT;
54          else return s2.top();
55      }
56
57  private:
58      stack s1, s2;
59  };
```

```
1   #include <iostream>
2   using namespace std;
3
4   const int MAX_INT = ~(1<<31); // 2147483647
5
6   typedef struct node {
7       int val;
8       int min;
9   } node;
10
11  class StackWithMin{
12  private:
13      node* buf;
14      int curr;
15  public:
16      StackWithMin(int size = 1000){
17          buf = new node[size];
18          buf[0].min = MAX_INT;
19          curr = 0;
20      }
21      ~StackWithMin(){
22          delete[] buf;
23      }
24
25      void push(int val) {
26          buf[++curr].val = val;
27          if ( val < buf[curr-1].min )
28              buf[curr].min = val;
29          else
30              buf[curr].min = buf[curr-1].min;
31      }
32      void pop() {
33          --curr;
34      }
35      int top() {
36          return buf[curr].val;
37      }
```

```
38      bool empty() {
39          //return (buf[curr].min == MAX_INT);
40          return curr == 0;  // much more easier
41      }
42      int min() {
43          return buf[curr].min;
44      }
45  };
46
47
48  // maintain two stack to save space
49  class stack {
50   private:
51      int *buf;
52      int curr;
53   public:
54      stack(int size = 1000) {
55          buf = new int[size];
56          curr = -1;
57      }
58      ~stack() {
59          delete[] buf;
60      }
61
62      void push(int val) {
63          buf[++curr] = val;
64      }
65      void pop() {
66          --curr;
67      }
68      int top() {
69          return buf[curr];
70      }
71      bool empty() {
72          return curr == -1;
73      }
74  };
75
76  class StackWithMin1{
77  private:
78      stack s1, s2;
79  public:
80      StackWithMin1(){
81      }
82      ~StackWithMin1(){
83      }
84
85      void push(int val) {
86          s1.push(val);
87          // if ( val <= s2.top() )
88          if ( val <= min() )  // more direct
89              s2.push(val);
90      }
91      void pop() {
92          if ( s1.top() == min() )
93              s2.pop();
94          int val = s1.top();
95          s1.pop();
96      }
```

```
97      int top() {
98          return s1.top();
99      }
100     bool empty() {
101         return s1.empty();
102     }
103     int min() {
104         if ( s2.empty() )
105             return MAX_INT;
106         else
107             return s2.top();
108     }
109 };
110
111
112 int main() {
113
114     StackWithMin1 mystack;
115
116     for(int i=0; i<20; ++i)
117         mystack.push(i);
118     cout << mystack.min() << "␣" << mystack.top() << endl;
119     mystack.push(-50);
120     mystack.push(-100);
121     cout << mystack.min() << "␣" << mystack.top() << endl;
122     mystack.pop();
123     cout << mystack.min() << "␣" << mystack.top() << endl;
124
125     return 0;
126 }
```

3.3 Imagine a (literal) stack of plates. If the stack gets too high, it might topple. Therefore, in real life, we would likely start a new stack when the previous stack exceeds some threshold. Implement a data structure SetOfStacks that mimics this. SetOfStacks should be composed of several stacks, and should create a new stack once the previous one exceeds capacity. SetOfStacks.push() and SetOfStacks.pop() should behave identically to a single stack (that is, pop() should return the same values as it would if there were just a single stack).

FOLLOW UP

Implement a function popAt(int index) which performs a pop operation on a specific sub-stack.

()SetOfStacks SetOfStacks SetOfStacks.push()SetOfStacks.pop()

popAt(int index)pop

Solution:

popAtSetOfStacks SetOfStackscur push cur1pop cur1top SetOfStacks pushpop

```
1  class SetOfStacks{//without popAt()
2  private:
3      stack *st;
4      int cur;
5      int capacity;
6
7  public:
8      SetOfStacks(int capa=STACK_NUM){
9          st = new stack[capa];
10         cur = 0;
11         capacity = capa;
12     }
13     ~SetOfStacks(){
14         delete[] st;
```

```
15      }
16      void push(int val){
17          if(st[cur].full()) ++cur;
18          st[cur].push(val);
19      }
20      void pop(){
21          if(st[cur].empty()) --cur;
22          st[cur].pop();
23      }
24      int top(){
25          if(st[cur].empty()) --cur;
26          return st[cur].top();
27      }
28      bool empty(){
29          if(cur==0) return st[0].empty();
30          else return false;
31      }
32      bool full(){
33          if(cur==capacity-1) return st[cur].full();
34          else return false;
35      }
36  };
```

popAt popAtpopAt popAt cur popAtpopAt cur()push popcur poppopAttopempty

```
1   class SetOfStacks1{
2   private:
3       stack *st;
4       int cur;
5       int capacity;
6
7   public:
8       SetOfStacks1(int capa=STACK_NUM){
9           st = new stack[capa];
10          cur = 0;
11          capacity = capa;
12      }
13      ~SetOfStacks1(){
14          delete[] st;
15      }
16      void push(int val){
17          if(st[cur].full()) ++cur;
18          st[cur].push(val);
19      }
20      void pop(){
21          while(st[cur].empty()) --cur;
22          st[cur].pop();
23      }
24      void popAt(int idx){
25          while(st[idx].empty()) --idx;
26          st[idx].pop();
27      }
28      int top(){
29          while(st[cur].empty()) --cur;
30          return st[cur].top();
31      }
32      bool empty(){
33          while(cur!=-1 && st[cur].empty()) --cur;
34          if(cur==-1) return true;
```

```
35          else return false;
36      }
37      bool full(){
38          if(cur==capacity-1) return st[cur].full();
39          else return false;
40      }
41  };
```

```
1   #include <iostream>
2   using namespace std;
3
4   const int STACK_SIZE = 100;
5   const int STACK_NUM = 10;
6
7   class stack {
8    private:
9       int *buf;
10      int curr;
11      int capacity;
12   public:
13      stack(int capa = STACK_SIZE) {
14          buf = new int[capa];
15          curr = -1;
16          capacity = capa;
17      }
18      ~stack() {
19          delete[] buf;
20      }
21
22      void push(int val) {
23          buf[++curr] = val;
24      }
25      void pop() {
26          --curr;
27      }
28      int top() {
29          return buf[curr];
30      }
31      bool empty() {
32          return curr == -1;
33      }
34      bool full() {
35          return curr == capacity-1;
36      }
37  };
38
39  class SetOfStacks {  // without popAt() yet
40  private:
41      stack *st;
42      int curr;
43      int capacity;
44  public:
45      SetOfStacks(int capa = STACK_NUM) {
46          st = new stack[capa];
47          curr = 0;
48          capacity = capa;
49      }
50      ~SetOfStacks() {
51          delete[] st;
```

```
52      }
53
54      void push(int val) {
55          if ( st[curr].full() )
56              ++curr;
57          st[curr].push(val);
58      }
59      void pop() {
60          if ( st[curr].empty() )
61              --curr;
62          st[curr].pop();
63      }
64      int top() {
65          if ( st[curr].empty() )
66              --curr;
67          return st[curr].top();
68      }
69      bool empty() {
70          if ( curr == 0 )
71              return  st[curr].empty() ;
72          else
73              return false;
74      }
75      bool full() {
76          if ( curr == capacity-1 )
77              return st[curr].full();
78          else
79              return false;
80      }
81  };
82
83
84  // with popAt() function
85  class SetOfStacks1 {
86  private:
87      stack *st;
88      int curr;
89      int capacity;
90  public:
91      SetOfStacks1(int capa = STACK_NUM) {
92          st = new stack[capa];
93          curr = 0;
94          capacity = capa;
95      }
96      ~SetOfStacks1() {
97          delete[] st;
98      }
99
100     void push(int val) {
101         if ( st[curr].full() )
102             ++curr;
103         st[curr].push(val);
104     }
105     void pop() {
106         while ( st[curr].empty() && curr > 0 ) // consider first stack empty condition
107             --curr;
108         st[curr].pop();
109     }
110     void popAt(int val) {
```

```
111             if ( st[val].empty() && val > 0 )
112                 --val;
113             st[val].pop();
114             if ( curr == val && st[val].empty() ) // I think I need to consider this condition
115                 --curr;
116         }
117         int top() {
118             while ( st[curr].empty() && curr > 0 )
119                 --curr;
120             return st[curr].top();
121         }
122         bool empty() {
123             //while ( st[curr].empty() && curr > 0 ) --curr;
124             //if ( curr == 0 ) return   st[curr].empty() ;
125 
126             while ( curr != -1 && st[curr].empty() )
127                 --curr;
128             if (curr == -1) return true;
129             else
130                 return false;
131         }
132         bool full() {
133             if ( curr == capacity-1 )
134                 return st[curr].full();
135             else
136                 return false;
137         }
138     };
139 
140 
141     int main() {
142         SetOfStacks1 s1;
143         for (int i = 0; i < 3*STACK_SIZE+1; ++i )
144             s1.push(i);
145         for (int i = 0; i < STACK_SIZE; ++i ) {
146             s1.popAt(0);
147             s1.popAt(2);
148         }
149         s1.popAt(3);
150         while ( !s1.empty() ) {
151             cout << s1.top() << endl;
152             s1.pop();
153         }
154         return 0;
155     }
```

3.4 In the classic problem of the Towers of Hanoi, you have 3 rods and N disks of different sizes which can slide onto any tower. The puzzle starts with disks sorted in ascending order of size from top to bottom (e.g., each disk sits on top of an even larger one). You have the following constraints:

(A) Only one disk can be moved at a time.

(B) A disk is slid off the top of one rod onto the next rod.

(C) A disk can only be placed on top of a larger disk.

Write a program to move the disks from the first rod to the last using Stacks.

(Google)

Solution:

void hanoi(int n, char src, char bri, char dst);

- nsrcdstbri(bridge)
- n1n
-

OK hanoi (srcbridst) (1 n, 0, 0)src1nn (0, 0, 1 n)dst1nn ndst (n, 1 n-1, 0)n dst ()hanoi

- (1 n, 0, 0)
- (n, 1 n-1, 0)

hanoi(n-1, src, dst, bri)n-1 srcbridst
nsrcdst
cout«"Move disk "«n«" from "«src«" to "«dst«endl;

(0, 1 n-1, n)
hanoin-1bridstsrc hanoi(n-1, bri, src, dst)
(0, 0, 1 n)

- hanoi(n-1, src, dst, bri);
- cout«"Move disk "«n«" from "«src«" to "«dst«endl;
- hanoi(n-1, bri, src, dst);

n1 srcdst
if(n==1)
cout«"Move disk "«n«" from "«src«" to "«dst«endl;

```
#include <iostream>
using namespace std;

void hanoi(int n, char src, char bri, char dst){
    if(n==1){
        cout<<"Move disk "<<n<<" from "<<src<<" to "<<dst<<endl;
    }
    else{
        hanoi(n-1, src, dst, bri);
        cout<<"Move disk "<<n<<" from "<<src<<" to "<<dst<<endl;
        hanoi(n-1, bri, src, dst);
    }
}

int main(){
    int n = 3;
    hanoi(n, 'A', 'B', 'C');
    return 0;
}
```

```
struct op{
    int begin, end;
    char src, bri, dst;
    op(){
    }
    op(int pbegin, int pend, int psrc, int pbri, int pdst):begin(pbegin), end(pend), src
    }
};
```

5srcbeginend srcdstbriendn srcbridstbegin beginend

- (1 n, 0, 0)
- (n, 1 n-1, 0)
- (0, 1 n-1, n)
- (0, 0, 1 n)

stack<op> st;
st.push(op(1, n, src, bri, dst));
src1 ndstbri stbeginendpush (push) push() beginend hanoi

```
void hanoi(int n, char src, char bri, char dst){
    stack<op> st;
    op tmp;
    st.push(op(1, n, src, bri, dst));
    while(!st.empty()){
        tmp = st.top();
        st.pop();
        if(tmp.begin != tmp.end){
            st.push(op(tmp.begin, tmp.end-1, tmp.bri, tmp.src, tmp.dst));
            st.push(op(tmp.end, tmp.end, tmp.src, tmp.bri, tmp.dst));
            st.push(op(tmp.begin, tmp.end-1, tmp.src, tmp.dst, tmp.bri));
        }
        else{
            cout<<"Move disk "<<tmp.begin<<" from "<<tmp.src<<" to "<<tmp.dst<<endl;
        }

    }
}
```

```
#include <iostream>
using namespace std;

void hanoi(int n, string src, string bri, string dst) {
    if (n == 1)
        cout << "Move disk " << n << " from " << src << " to " << dst << endl;
    else {
        hanoi(n-1, src, dst, bri);
        cout << "Move disk " << n << " from " << src << " to " << dst << endl;
        hanoi(n-1, bri, src, dst);
    }
}

int main() {
    int n = 3;
    hanoi(n, "src", "bri", "dst");
    return 0;
}
```

```
#include <iostream>
#include <stack>
using namespace std;

struct op {
    int begin, end;
    char src, bri, dst;
    op() {
```

```
9       }
10      op(int pbegin,int pend,int psrc,int pbri,int pdst):begin(pbegin),end(pend),src(psrc)
11      }
12  };
13
14  void hanoi(int n, char src, char bri, char dst) {
15      stack<op> st;
16      op tmp;
17      st.push( op(1, n, src, bri, dst) );
18      while ( !st.empty() ) {
19          tmp = st.top();
20          st.pop();
21          if (tmp.begin != tmp.end) {
22              st.push( op(tmp.begin, tmp.end-1, tmp.bri, tmp.src, tmp.dst) );
23              st.push( op(tmp.end, tmp.end, tmp.src, tmp.bri, tmp.dst) );
24              st.push( op(tmp.begin, tmp.end-1, tmp.src, tmp.dst, tmp.bri) );
25          } else {
26              cout << "Move␣disk␣" << tmp.begin << "␣from␣" << tmp.src << "␣to␣" << tmp.ds
27          }
28      }
29  }
30
31  int main() {
32      int n = 3;
33      hanoi(n, 'A', 'B', 'C');
34      return 0;
35  }
```

3.5 Implement a MyQueue class which implements a queue using two stacks.

MyQueue

Solution:

(FIFO)(FILO)

```
1   template <typename T>
2   class MyQueue{
3   public:
4       MyQueue(){
5
6       }
7       ~MyQueue(){
8
9       }
10      void push(T val){
11          move(sout, sin);
12          sin.push(val);
13      }
14      void pop(){
15          move(sin, sout);
16          sout.pop();
17      }
18      T front(){
19          move(sin, sout);
20          return sout.top();
21      }
22      T back(){
23          move(sout, sin);
24          return sin.top();
25      }
26      int size(){
27          return sin.size()+sout.size();
```

```
28      }
29      bool empty(){
30          return sin.empty()&&sout.empty();
31      }
32      void move(stack<T> &src, stack<T> &dst){
33          while(!src.empty()){
34              dst.push(src.top());
35              src.pop();
36          }
37      }
38
39  private:
40      stack<T> sin, sout;
41  };
```

```
1   template <typename T>
2   class MyQueue1{
3   public:
4       MyQueue1(){
5
6       }
7       ~MyQueue1(){
8
9       }
10      void push(T val){
11          sin.push(val);
12      }
13      void pop(){
14          move(sin, sout);
15          sout.pop();
16      }
17      T front(){
18          move(sin, sout);
19          return sout.top();
20      }
21      T back(){
22          move(sout, sin);
23          return sin.top();
24      }
25      int size(){
26          return sin.size()+sout.size();
27      }
28      bool empty(){
29          return sin.empty()&&sout.empty();
30      }
31      void move(stack<T> &src, stack<T> &dst){
32          if(dst.empty()){
33              while(!src.empty()){
34                  dst.push(src.top());
35                  src.pop();
36              }
37          }
38      }
39
40  private:
41      stack<T> sin, sout;
42  };
```

```
#include <iostream>
#include <stack>
using namespace std;

template <typename T>
class MyQueue {
private:
    stack<T> sin, sout;
public:
    MyQueue() {
    }
    ~MyQueue() {
    }

    void push(T val) {
        move(sout, sin);
        sin.push(val);
    }
    void pop() {
        move(sin, sout);
        sout.pop();
    }
    T front() {
        move(sin, sout);
        sout.top();
    }
    T back() {
        move(sout, sin);
        sin.top();
    }
    int size() {
        return sin.size() + sout.size();
    }
    bool empty() {
        return sin.empty() && sout.empty();
    }
    void move(stack<T> &src, stack<T> &dst) {
        while ( !src.empty() ) {
            dst.push( src.top() );
            src.pop();
        }
    }
};


// control move under different member functions
template <typename T>
class MyQueue1 {
private:
    stack<T> sin, sout;
public:
    MyQueue1() {
    }
    ~MyQueue1() {
    }

    void push(T val) {
        sin.push(val);
```

```
59      }
60      void pop() {
61          if ( sout.empty() )
62              move(sin, sout);
63          sout.pop();
64      }
65      T front() {
66          if ( sout.empty() )
67              move(sin, sout);
68          sout.top();
69      }
70      T back() {
71          if ( sin.empty() )
72              move(sout, sin);
73          sin.top();
74      }
75      int size() {
76          return sin.size() + sout.size();
77      }
78      bool empty() {
79          return sin.empty() && sout.empty();
80      }
81      void move(stack<T> &src, stack<T> &dst) {
82          while ( !src.empty() ) {
83              dst.push( src.top() );
84              src.pop();
85          }
86      }
87  };
88
89
90  // control under move function only
91  template <typename T>
92  class MyQueue2 {
93  private:
94      stack<T> sin, sout;
95  public:
96      MyQueue2() {
97      }
98      ~MyQueue2() {
99      }
100
101     void push(T val) {
102         sin.push(val);
103     }
104     void pop() {
105         move(sin, sout);
106         sout.pop();
107     }
108     T front() {
109         move(sin, sout);
110         sout.top();
111     }
112     T back() {
113         move(sout, sin);
114         sin.top();
115     }
116     int size() {
117         return sin.size() + sout.size();
```

```
118         }
119         bool empty() {
120             return sin.empty() && sout.empty();
121         }
122         void move(stack<T> &src, stack<T> &dst) {
123             if ( dst.empty() ) {
124                 while ( !src.empty() ) {
125                     dst.push( src.top() );
126                     src.pop();
127                 }
128             }
129         }
130     };
131
132     int main(){
133         MyQueue<int> q;
134         MyQueue1<int> q1;
135
136         for(int i=0; i<10; ++i){
137             q.push(i);
138             q1.push(i);
139         }
140
141         cout<<q.front()<<" "<<q.back()<<endl;
142         cout<<q1.front()<<" "<<q1.back()<<endl;
143         cout<<endl;
144         q.pop();
145         q1.pop();
146         q.push(10);
147         q1.push(10);
148         cout<<q.front()<<" "<<q.back()<<endl;
149         cout<<q1.front()<<" "<<q1.back()<<endl;
150         cout<<endl;
151         cout<<q.size()<<" "<<q.empty()<<endl;
152         cout<<q1.size()<<" "<<q1.empty()<<endl;
153         return 0;
154     }
```

3.6 Write a program to sort a stack in ascending order. You should not make any assumptions about how the stack is implemented. The following are the only functions that should be used to write this program: push | pop | peek | isEmpty.

push | pop | peek | isEmpty

Solution:

1

```
1   stack<int> Ssort(stack<int> s){
2       stack<int> t;
3       while(!s.empty()){
4           int data = s.top();
5           s.pop();
6           while(!t.empty() && t.top()>data){
7               s.push(t.top());
8               t.pop();
9           }
10          t.push(data);
11      }
12      return t;
13  }
```

2

```
void Qsort(stack<int> &s){
    priority_queue< int,vector<int>,greater<int> > q;
    while(!s.empty()){
        q.push(s.top());
        s.pop();
    }
    while(!q.empty()){
        s.push(q.top());
        q.pop();
    }
}
```

```
#include <iostream>
#include <stack>
#include <queue>
#include <cstdlib>
using namespace std;

stack<int> Ssort(stack<int> s) {
    stack<int> t;
    while ( !s.empty() ) {
        int data = s.top();
        s.pop();
        while ( !t.empty() && t.top() > data ) {
            s.push( t.top() );
            t.pop();
        }
        t.push(data);
    }
    return t;
}

void Qsort(stack<int> &s) {
    priority_queue< int,vector<int>,greater<int> > q;
    while ( !s.empty() ) {
        q.push( s.top() );
        s.pop();
    }
    while ( !q.empty() ) {
        s.push( q.top() );
        q.pop();
    }
};

int main() {
    srand( (unsigned)time(0) );
    stack<int> s;
    for (int i = 0; i < 10; ++i)
        s.push( rand()%100 );
    Qsort(s);
    while ( !s.empty() ) {
        cout << s.top() << endl;
        s.pop();
    }
```

```
43         return 0;
44     }
```

Chapter 4

Trees and Graphs

4.1 Implement a function to check if a tree is balanced. For the purposes of this question, a balanced tree is defined to be a tree such that no two leaf nodes differ in distance from the root by more than one.

1

Solution:

1 1 (fl2)

1

```
int d = 0, num = 0, dep[maxn];
void getDepth(Node *head){
    if(head == NULL) return;
    ++d;
    getDepth(head->lchild);
    if(head->lchild == NULL && head->rchild == NULL)
        dep[num++] = d;
    getDepth(head->rchild);
    --d;
}
```

1

```
bool isBalance(Node *head){
    if(head == NULL) return true;
    getDepth(head);
    int max = dep[0], min = dep[0];
    for(int i=0; i<num; ++i){
        if(dep[i]>max) max = dep[i];
        if(dep[i]<min) min = dep[i];
    }
    if(max-min > 1) return false;
    else return true;
}
```

```
#include <iostream>
#include <cstring>
using namespace std;

const int maxn = 100;

struct Node {
    int key;
```

```
9       Node *left, *right, *parent;
10  };
11  Node *head, *p, node[maxn];
12  int cnt;
13
14  void init() {
15      head = p = NULL;
16      memset(node, '\0', sizeof(node));
17      cnt = 0;
18  }
19
20  void insert(Node* &head, int x) {
21      if (head == NULL) {
22          node[cnt].key = x;
23          node[cnt].parent = p;
24          head = &node[cnt++];
25          return;
26      }
27      p = head;
28      if (x < head->key)
29          insert(head->left, x);
30      else
31          insert(head->right, x);
32  }
33
34  int d = 0, num = 0, depth[maxn];
35  void getDepth(Node * head) {
36      if (head == NULL) return;
37      ++d;
38      getDepth(head->left);
39      if ( !head->left && !head->right ) // leaf node
40          depth[num++] = d;              // leaf node depth
41      getDepth(head->right);
42      --d;
43  }
44
45  bool isBalanced(Node* head) {
46      if (head == NULL) return true;
47      num = 0;
48      getDepth(head);
49
50      int min = depth[0];
51      int max = depth[0];
52      for (int i = 1; i < num; ++i) {
53          if ( depth[i] < min ) min = depth[i];
54          if (depth[i] > max) max = depth[i];
55      }
56      if (max - min > 1)
57          return false;
58      else
59          return true;
60  }
61
62  int main() {
63      init();
64      int a[] = {5, 3, 8, 1, 4, 7, 10, 2, 6, 9, 11, 12};
65      for (int i = 0; i < 12; ++i )
66          insert(head, a[i]);
67      cout << isBalanced(head) << endl;
```

```
68      return 0;
69  }
```

4.2 Given a directed graph, design an algorithm to find out whether there is a route between two nodes.

Solution:

(BFS)

```
1   bool route(int src, int dst){
2       q.push(src);
3       visited[src] = true;
4       while(!q.empty()){
5           int t = q.front();
6           q.pop();
7           if(t == dst) return true;
8           for(int i=0; i<n; ++i)
9               if(g[t][i] && !visited[i]){
10                  q.push(i);
11                  visited[i] = true;
12              }
13      }
14      return false;
15  }
```

```
1   #include <iostream>
2   #include <cstring>
3   #include <queue>
4   #include <fstream>
5   using namespace std;
6
7   const int maxn = 100;
8   bool g[maxn][maxn], visited[maxn];
9   int n;
10  queue<int> q;
11
12  void init() {
13      memset(g, false, sizeof(g));
14      memset(visited, false, sizeof(visited));
15  }
16
17  bool route(int src, int dst) {
18      q.push(src);
19      visited[src] = true;
20      while ( !q.empty() ) {
21          int t = q.front();
22          q.pop();
23          if (t == dst) return true;
24
25          for (int i = 0; i < n; ++i )
26              if ( g[t][i] && !visited[i] ) {
27                  q.push(i);
28                  visited[i] = true;
29              }
30      }
31      return false;
32  }
33
```

```
int main() {
    freopen("4.2.in", "r", stdin);
    init();
    int m, u, v;
    cin >> n >> m;
    for (int i = 0; i < m; ++i) {
        cin >> u >> v;
        g[u][v] = true;
    }
    cout << route(0, 6) << endl;

    fclose(stdin);

    return 0;
}
```

4.3 Given a sorted (increasing order) array, write an algorithm to create a binary tree with minimal height.
()
Solution:

```
void create_minimal_tree(Node* &head, Node *parent, int a[], int start, int end){
    if(start <= end){
        int mid = (start + end)>>1;
        node[cnt].key = a[mid];
        node[cnt].parent = parent;
        head = &node[cnt++];
        create_minimal_tree(head->lchild, head, a, start, mid-1);
        create_minimal_tree(head->rchild, head, a, mid+1, end);
    }
}
```

```
#include <iostream>
#include <cstring>
using namespace std;

const int maxn = 100;

struct Node {
    int key;
    Node *left, *right;
    Node *parent;
};
Node *p, node[maxn];
int cnt;

void init() {
    p = NULL;
    memset(node, '\0', sizeof(node));
    cnt = 0;
}

void create_minimal_tree(Node* &head, Node *parent, int a[], int start, int end) {
    if (start <= end) {
        int mid = (start + end) / 2;
        node[cnt].key = a[mid];
        node[cnt].parent = parent;
        head = &node[cnt++];
```

```
27          create_minimal_tree(head->left , head, a, start , mid-1);
28          create_minimal_tree(head->right , head, a, mid+1, end);
29      }
30  }
31
32  int height(Node* head) {
33      if (head == NULL) return 0;
34      return max( height(head->left), height(head->right) ) + 1;
35  }
36
37  int main() {
38      init();
39      int a[] = {0, 1, 2, 3, 4, 5, 6, 7, 8};
40      Node* head = NULL;
41      create_minimal_tree(head, NULL, a, 0, 8);
42      cout << height(head) << endl;
43      return 0;
44  }
```

4.4 Given a binary search tree, design an algorithm which creates a linked list of all the nodes at each depth (i.e., if you have a tree with depth D, youll have D linked lists).

(DD)

Solution:

BFSi i+1 Nodelist<Node*> vector

```
1  vector<list<Node*> > find_level_linklists(Node *head){
2      vector<list<Node*> > res;
3      int level = 0;
4      list<Node*> li;
5      li.push_back(head);
6      res.push_back(li);
7      while(!res[level].empty()){
8          list<Node*> l;
9          list<Node*>::iterator it;
10         for(it=res[level].begin(); it!=res[level].end(); ++it){
11             Node *n = *it;
12             if(n->lchild) l.push_back(n->lchild);
13             if(n->rchild) l.push_back(n->rchild);
14         }
15         ++level;
16         res.push_back(l);
17     }
18     return res;
19 }
```

```
1  #include <iostream>
2  #include <cstring>
3  #include <vector>
4  #include <list>
5  using namespace std;
6
7  const int maxn = 100;
8
9  struct Node {
10     int key;
11     Node *left , *right , *parent;
12 };
```

```
13 Node *p, node[maxn];
14 int cnt;
15
16 void init() {
17     p = NULL;
18     memset(node, '\0', sizeof(node));
19     cnt = 0;
20 }
21
22 void createMinimalTree (Node* &head, Node *parent, int a[], int start, int end) {
23     if (start <= end) {
24         //int mid = (start + end) / 2;
25         int mid = (start + end) >> 1; // a cheaper implementation
26         node[cnt].key = a[mid];
27         node[cnt].parent = head;
28         head = &node[cnt++]; // needs special attention
29         createMinimalTree(head->left, head, a, start, mid-1);
30         createMinimalTree(head->right, head, a, mid+1, end);
31     }
32 }
33
34 vector<list<Node*> > find_level_linklists( Node *head ) {
35     vector<list<Node*> > res;
36     int level = 0;
37     list<Node*> li;
38     li.push_back(head);
39     res.push_back(li);
40
41     while ( !res[level].empty() ) {
42         list<Node*> l;
43         list<Node*>::iterator it;
44         for (it = res[level].begin(); it != res[level].end(); ++it) {
45             Node *n = *it;
46             if (n->left) l.push_back(n->left);
47             if (n->right) l.push_back(n->right);
48         }
49         ++level;
50         res.push_back(l); // mistake here before, no index
51     }
52     return res;
53 }
54
55 void print(vector<list<Node*> > res) {
56     vector< list<Node*> >::iterator vit;
57     for (vit = res.begin(); vit != res.end(); ++vit) {
58         list<Node*> li = *vit;
59         list<Node*>::iterator lit;
60         for (lit = li.begin(); lit != li.end(); ++lit) {
61             Node *n = *lit;
62             cout << n->key << '\t';
63         }
64         cout << endl;
65     }
66 }
67
68 int main() {
69     init();
70     int a[] = {0, 1, 2, 3, 4, 5, 6, 7, 8};
71     Node *head = NULL;
```

```
72      createMinimalTree(head, NULL, a, 0, 8);
73      vector<list<Node*> > res;
74      res = find_level_linklists(head);
75      print(res);
76      return 0;
77  }
```

4.5 Write an algorithm to find the next node (i.e., in-order successor) of a given node in a binary search tree where each node has a link to its parent.

()

Solution:

( )

```
1   Node* minimal(Node* no){
2       if(no == NULL) return NULL;
3       while(no->lchild)
4           no = no->lchild;
5       return no;
6   }
7   Node* successor(Node* no){
8       if(no == NULL) return NULL;
9       if(no->rchild) return minimal(no->rchild);
10      Node *y = no->parent;
11      while(y && y->rchild==no){
12          no = y;
13          y = y->parent;
14      }
15      return y;
16  }
```

```
1   #include <iostream>
2   #include <cstring>
3   using namespace std;
4
5   const int maxn = 100;
6   struct Node {
7       int key;
8       Node *left, *right, *parent;
9   };
10  Node* p, node[maxn];
11  int cnt;
12
13  void init() {
14      p = NULL;
15      memset(node, '\0', sizeof(node));
16      cnt = 0;
17  }
18
19  void create_minimal_tree(Node* &head, Node *parent, int a[], int start, int end) {
20      if (start <= end) {
21          int mid = (start + end) >> 1;
22          node[cnt].key = a[mid];
23          node[cnt].parent = head;
24          head = &node[cnt++];
25          create_minimal_tree(head->left, head, a, start, mid-1);
26          create_minimal_tree(head->right, head, a, mid+1, end);
27      }
```

```
28 }
29
30 Node* minimal(Node* head) {
31     if (head == NULL) return NULL;
32     while (head->left)
33         head = head->left;
34     return head;
35 }
36
37 Node* successor (Node* ptr) {
38     if (ptr == NULL) return NULL;
39     if (ptr->right)
40         return minimal(ptr->right);
41
42     Node* p = ptr->parent;
43     while (p && p->right == ptr) {
44         ptr = p;
45         p = p->parent;
46     }
47     return p;
48 }
49
50 int main() {
51     int a[] = {0, 1, 2, 3, 4, 5, 6, 7, 8};
52     init();
53     Node *head = NULL;
54     create_minimal_tree(head, NULL, a, 0, 8);
55     cout << "the head is " << head->key << endl;
56     cout << "the successor of head is " << (successor(head))->key << endl;
57     return 0;
58 }
```

4.6 Design an algorithm and write code to find the first common ancestor of two nodes in a binary tree. Avoid storing additional nodes in a data structure. NOTE: This is not necessarily a binary search tree.

Solution:

Avoid storing additional nodes in a data structure () Node Anyway

()

map()

```
1  Node* first_ancestor(Node* n1, Node* n2){
2      if(n1 == NULL || n2 == NULL) return NULL;
3      map<Node*, bool> m;
4      while(n1){
5          m[n1] = true;
6          n1 = n1->parent;
7      }
8      while(n2 && !m[n2]){
9          n2 = n2->parent;
10     }
11     return n2;
12 }
```

map

```
1  bool father(Node* n1, Node* n2){
2      if(n1 == NULL) return false;
3      else if(n1 == n2) return true;
4      else return father(n1->lchild, n2) || father(n1->rchild, n2);
5  }
```

```
Node* first_ancestor1(Node* n1, Node* n2){
    if(n1 == NULL || n2 == NULL) return NULL;
    while(n1){
        if(father(n1, n2)) return n1;
        n1 = n1->parent;
    }
    return NULL;
}
```

Node

```
void first_ancestor2(Node* head, Node* n1, Node* n2, Node* &ans){
    if(head==NULL || n1==NULL || n2==NULL) return;
    if(head && father(head, n1) && father(head, n2)){
        ans = head;
        first_ancestor2(head->lchild, n1, n2, ans);
        first_ancestor2(head->rchild, n1, n2, ans);
    }
}
```

ans ans

```
#include <iostream>
#include <cstring>
#include <map>
using namespace std;

const int maxn = 100;
struct Node {
    int key;
    Node *left, *right, *parent;
};
Node *p, node[maxn];
int cnt;

void init() {
    p = NULL;
    memset(node, '\0', sizeof(node));
    cnt = 0;
}

void create_minimal_tree(Node* &head, Node *parent, int a[], int start, int end){
    if(start <= end){
        int mid = (start + end)>>1;
        node[cnt].key = a[mid];
        node[cnt].parent = parent;
        head = &node[cnt++];
        create_minimal_tree(head->left, head, a, start, mid-1);
        create_minimal_tree(head->right, head, a, mid+1, end);
    }
}

Node* first_ancestor(Node *n1, Node *n2) {
    if (n1 == NULL || n2 == NULL) return NULL;
    map<Node*, bool> m;
    while (n1) {
        m[n1] = true;
        n1 = n1->parent;
    }
    while (n2 && !m[n2])
```

```
39              n2 = n2->parent;
40      return n2;
41  }
42
43  // mine, confusing, should be wrong
44  /*
45  Node* first_ancestor1(Node *n1, Node *n2) {
46      if (n1 == NULL || n2 == NULL) return NULL;
47      Node *p = n2;
48
49      while (n1) {
50          while (n2 && n2 != n1)
51              n2 = n2->parent;
52          n2 = p;
53          n1 = n1->parent;
54      }
55      return n2;
56  }  */
57
58
59  bool father(Node *n1, Node *n2) {
60      if (n1 == NULL) return false;
61      else if (n1 == n2) return true;
62      else return father(n1->left, n2) || father(n1->right, n2);
63  }
64
65  Node* first_ancestor2(Node *n1, Node *n2) {
66      if (n1 == NULL || n2 == NULL) return NULL;
67
68      while (n1) {
69          if (father(n1, n2)) return n1;
70          n1 = n1->parent;
71      }
72      return NULL;
73  }
74
75  // no parent pointer
76  void first_ancestor3(Node *head, Node *n1, Node *n2, Node* &ans) {
77      if (head == NULL || n1 == NULL || n2 == NULL) return;
78      if ( head && father(head, n1) && father(head, n2) ) {
79          ans = head;
80          first_ancestor3(head->left, n1, n2, ans);
81          first_ancestor3(head->right, n1, n2, ans);
82      }
83  }
84
85  Node* search(Node* head, int x) {
86      if (head == NULL) return NULL;
87      else if (x == head->key) return head;
88      else if (x <= head->key) return search(head->left, x);
89      else return search(head->right, x);
90  }
91
92  int main() {
93      init();
94      int a[] = {0, 1, 2, 3, 4, 5, 6, 7, 8};
95      Node *head = NULL;
96      create_minimal_tree(head, NULL, a, 0, 8);
97      Node *n1 = search(head, 3);
```

```
98      Node *n2 = search(head, 7);
99      cout << "keys: " << n1->key << " " << n2->key << endl;
100
101     Node* ans = first_ancestor2(n1, n2);
102     cout << "ans->key: " << ans->key << endl;
103
104     Node *ans1 = NULL;
105     first_ancestor3(head, n1, n2, ans1);
106     cout << "ans->key: " << ans->key << endl;
107
108     return 0;
109 }
```

4.7 You have two very large binary trees: T1, with millions of nodes, and T2, with hundreds of nodes. Create an algorithm to decide if T2 is a subtree of T1

T1T2T2T1

Solution:

T1T2

T1T2 T1T2T2 T1T1

```
1  bool match(Node* r1, Node* r2){
2      if(r1 == NULL && r2 == NULL) return true;
3      else if(r1 == NULL || r2 == NULL) return false;
4      else if(r1->key != r2->key) return false;
5      else return match(r1->lchild, r2->lchild) && match(r1->rchild, r2->rchild);
6  }
7  bool subtree(Node* r1, Node* r2){
8      if(r1 == NULL) return false;
9      else if(r1->key == r2->key){
10         if(match(r1, r2)) return true;
11     }
12     else return subtree(r1->lchild, r2) || subtree(r1->rchild, r2);
13 }
14 bool contain_tree(Node* r1, Node* r2){
15     if(r2 == NULL) return true;
16     else return subtree(r1, r2);
17 }
```

```
1  #include <iostream>
2  #include <cstring>
3  using namespace std;
4
5  const int maxn = 100;
6  struct Node {
7      int key;
8      Node *left, *right, *parent;
9  };
10 Node node[maxn];
11 int cnt;
12
13 void init() {
14     memset(node, '\0', sizeof(node));
15     cnt = 0;
16 }
17
18 void create_minimal_tree(Node * &head, Node *parent, int a[], int start, int end) {
19     if (start <= end ) {
```

```
20          int mid = (start + end) >> 1;
21          node[cnt].key = a[mid];
22          node[cnt].parent = parent;
23          head = &node[cnt++];
24          create_minimal_tree(head->left, head, a, start, mid-1);
25          create_minimal_tree(head->right, head, a, mid+1, end);
26      }
27  }
28
29  // if two tree matches
30  bool match(Node *n1, Node *n2) {
31      if (n1 == NULL && n2 == NULL) return true;
32      else if (n1 == NULL || n2 == NULL) return false;
33      else if (n1->key != n2->key) return false;
34      else
35          return match(n1->left, n2->left) && match(n1->right, n2->right);
36  }
37
38  bool subtree(Node *n1, Node *n2) {
39      if (n1 == NULL) return false;
40      else if (n1->key == n2->key) {
41          if ( match(n1, n2) ) return true;
42      }
43      else return subtree(n1->left, n2) || subtree(n1->right, n2);
44  }
45
46  bool contain_tree(Node *n1, Node *n2) {
47      if (n2 == NULL) return true;
48      else return subtree(n1, n2);
49  }
50
51  int main() {
52      init();
53      int a1[] = {0, 1, 2, 3, 4, 5, 6};
54      int a2[] = {0, 1, 2};
55      Node *r1 = NULL, *r2 = NULL;
56      create_minimal_tree(r1, NULL, a1, 0, 6);
57      create_minimal_tree(r2, NULL, a2, 0, 2);
58      if ( contain_tree(r1, r2) )
59          cout << "tree r1 contains tree r2" << endl;
60      else
61          cout  << "tree r1 does not contain tree r2" << endl;
62
63      return 0;
64  }
```

4.8 You are given a binary tree in which each node contains a value. Design an algorithm to print all paths which sum up to that value. Note that it can be any path in the tree - it does not have to start at the root.

Solution:

1 ( )sum

```
1  void find_sum(Node* head, int sum){
2      if(head == NULL) return;
3      Node *no = head;
4      int tmp = 0;
5      for(int i=1; no!=NULL; ++i){
6          tmp += no->key;
```

```
            if(tmp == sum)
                print(head, i);
            no = no->parent;
        }
        find_sum(head->lchild, sum);
        find_sum(head->rchild, sum);
    }
```

()

```
    void print(Node* head, int level){
        vector<int> v;
        for(int i=0; i<level; ++i){
            v.push_back(head->key);
            head = head->parent;
        }
        while(!v.empty()){
            cout<<v.back()<<" ";
            v.pop_back();
        }
        cout<<endl;
    }
```

2

```
    void print2(vector<int> v, int level){
        for(int i=level; i<v.size(); ++i)
            cout<<v.at(i)<<" ";
        cout<<endl;
    }
    void find_sum2(Node* head, int sum, vector<int> v, int level){
        if(head == NULL) return;
        v.push_back(head->key);
        int tmp = 0;
        for(int i=level; i>-1; --i){
            tmp += v.at(i);
            if(tmp == sum)
                print2(v, i);
        }
        vector<int> v1(v), v2(v);
        find_sum2(head->lchild, sum, v1, level+1);
        find_sum2(head->rchild, sum, v2, level+1);
    }
```

12

12level1level1 2level0

```
#include <iostream>
#include <vector>
#include <cstring>
using namespace std;

const int maxn = 100;
struct Node {
    int key;
    Node *left, *right, *parent;
};
Node node[maxn];
```

```
12 int cnt;
13
14 void init() {
15     memset(node, '\0', sizeof(node));
16     cnt = 0;
17 }
18
19 void create_minimal_tree(Node* &head, Node *parent, int a[], int start, int end) {
20     if (start <= end) {
21         int mid = (start + end) >> 1;
22         node[cnt].key = a[mid];
23         node[cnt].parent = parent;
24         head = &node[cnt++];
25         create_minimal_tree(head->left, head, a, start, mid-1);
26         create_minimal_tree(head->right, head, a, mid+1, end);
27     }
28 }
29
30 // method 1: with pointer pointing to parent
31 void print(Node *head, int level) {
32     vector<int> path;
33     for (int i = 0; i < level; ++i) {
34         path.push_back(head->key);
35         head = head->parent;
36     }
37
38     vector<int>::iterator it;
39     /*for (it = path.end(); it != path.begin(); --it)
40           cout << *it << endl;  */ // somewhere here still WRONG!!!
41     while ( !path.empty() ) {
42         cout << path.back() << " ";
43         path.pop_back();
44     }
45     cout << endl;
46 }
47
48 void find_sum(Node *head, int sum) {
49     if (head == NULL) return;
50
51     Node *ptr = head;
52     int val = 0;
53
54     for (int i = 1; ptr != NULL; ++i ) {
55         val += ptr->key;
56         if (val == sum)
57             print(head, i);
58         ptr = ptr->parent;
59     }
60     find_sum(head->left, sum);
61     find_sum(head->right, sum);
62 }
63
64
65 // method 2: without pointer pointing to parent, complicated
66 void print1(vector<int> v, int level) {
67     for (int i = level; i < v.size(); ++i)
68         cout << v.at(i) << " ";
69     cout << endl;
70 }
```

```
71
72  void find_sum1(Node *head, int sum, vector<int> v, int level) {
73      if (head == NULL) return;
74      v.push_back(head->key);
75
76      int val = 0;
77      for (int i = level; i > -1; --i ) {
78          val += v.at(i);
79          if (val == sum)
80              print1(v, i);
81      }
82      vector<int> v1(v), v2(v);
83      find_sum1(head->left, sum, v1, level+1);
84      find_sum1(head->right, sum, v2, level+1);
85  }
86
87
88  int main() {
89      init();
90      int a[] = {4, 3, 8, 5, 2, 1, 6};
91      Node *head = NULL;
92      create_minimal_tree(head, NULL, a, 0, 6);
93      //find_sum(head, 8);
94      vector<int> v;
95      find_sum1(head, 8, v, 0);
96      return 0;
97  }
```

Chapter 5

Bit Manipulation

5.1 You are given two 32-bit numbers, N and M, and two bit positions, i and j. Write a method to set all bits between i and j in N equal to M (e.g., M becomes a substring of N located at i and starting at j).

EXAMPLE:

Input: N = 10000000000, M = 10101, i = 2, j = 6

Output: N = 10001010100

32NMij NijM(MNNij)

: N = 10000000000, M = 10101, i = 2, j = 6

: N = 10001010100

Solution:

1N0i([0, i))ret N0j0([0, j])j+1j+1 0(m«i)ret

```
int update_bits(int n, int m, int i, int j){
    int ret = (1 << i) -1;
    ret &= n;
    return ((n>>(j+1)) << (j+1)) | (m<<i) | ret;
}
```

210(m) 1masknn0 mi

```
int update_bits1(int n, int m, int i, int j){
    int max = ~0;
    int left = max - ((1 << j+1) - 1);
    int right = ((1 << i) -1);
    int mask = left | right;
    return (n & mask) | (m << i);
}
```

C++

1i i1

```
int a = 1;
a <<= 31;
a >>= 31;
cout << a << endl;
```

1i i0

```
unsigned int a = 1;
a <<= 31;      //a:1310
a >>= 31;      //a:31011
cout<<a<<endl;      //1
```

0
01 0
int~(1«31)0311
int1«311310
unsigned int~0321
unsigned int0
int

```
#include <iostream>
#include <vector>
using namespace std;

void print_binary(int n) {
    vector<int> v;
    int len = 8*sizeof(int);  // bits count
    int mask = 1;
    while(len--) {
        if (n & mask) v.push_back(1);
        else v.push_back(0);

      //mask <<= 1;       // <<= equivalent !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
        mask = mask << 1;
    }
    int cnt = 0;
    while ( !v.empty() ) {
        if (cnt%4==0) cout << "␣";
        if (cnt%8==0) cout << "␣␣"; // for printing propose only

        cout << v.back();
        v.pop_back();

        ++cnt;
    }
    cout << endl;
}
/*
jenny@jenny-G50VT ~/docu/iv/ctci $ g++ one.cpp
jenny@jenny-G50VT ~/docu/iv/ctci $ ./a.out
   0000 0000   0000 0000   0000 0100   0000 0000
   0000 0000   0000 0000   0000 0000   0001 0101
   0000 0000   0000 0000   0000 0100   0101 0100
*/

int update_bits(int n, int m, int i, int j) {

    int ret = (1 << i) -1;
    ret &= n;

    /* not sure if this method works
    int ret = n;
    ret = ret << (32-i);
    ret = ret >> (32-i);
    */
    /*
    n = n >> (j+1);
    n = n << (j+1);
    n = n | (m << i);
    n = n | ret;
```

```
51      */
52      return ( (n >> (j+1)) << (j+1) ) | m << i | ret;
53  }
54
55  int update_bits2(int n, int m, int i, int j) {
56      int max = ~0; // 1 3232111111111 11111111 11111111 11111111
57      int left = max - ((1 << j+1)-1) ; // 11111111 11111000 00000000 00000000
58      int right = (1 << i) - 1;         // 00000000 00000000 00000000 00111111
59      int mask = left | right;          // 11111111 11111000 00000000 00111111
60      /*
61        int mask = (1 << (33-j) - 1) << j;
62        mask |= (1 << i)-1 ;
63      */
64      return (n & mask) | (m << i);
65  }
66
67  int main() {
68      int n = 1 << 10, m = 21;
69      int ans = update_bits(n, m, 2, 6);
70      print_binary(n);
71      print_binary(m);
72      print_binary(ans);
73      return 0;
74  }
```

5.2 Given a (decimal - e.g. 3.72) number that is passed in as a string, print the binary representation.If the number can not be represented accurately in binary, print ERROR.

(string) "ERROR"

Solution:

22 12 21 21111 00 32 32

```
1  #include <iostream>
2  #include <cstring>
3  #include <cstdlib>
4  using namespace std;
5
6  string print_binary(string val) {
7      int pos = val.find('.', 0);
8      int intpart =  atoi( val.substr(0, pos).c_str() ); // c_str() function
9      double decipart = atof( val.substr(pos, val.length()-pos).c_str() );
10     string left = "", right = "";
11
12     while(intpart > 0) {
13         if (intpart & 1) left = "1"+left;
14         else left = "0" + left;
15         intpart >>= 1;
16     }
17
18     int cnt = 0;
19     while (decipart > 0) {
20         if ( right.length() > 32 ) return "ERROR";
21
22         decipart *= 2;
23         if (decipart >= 1) {
24             right += "1";
25             decipart -= 1;
26         } else
27             right += "0";
```

```
28      }
29
30      return left + "." + right;
31  }
32
33  int main() {
34      string val = "19.25";
35      cout << print_binary(val) << endl;
36      return 0;
37  }
```

5.3 Given an integer, print the next smallest and next largest number that have the same number of 1 bits in their binary representation.

x1x xx

Solution:

x1num x11num x1 x1x1

```
1   int next(int x){
2       int max_int = ~(1<<31);
3       int num = count_one(x);
4       if(num == 0 || x == -1) return -1;
5       for(++x; count_one(x) != num && x < max_int; ++x);
6       if(count_one(x) == num) return x;
7       return -1;
8   }
9
10  int previous(int x){
11      int min_int = (1<<31);
12      int num = count_one(x);
13      if(num == 0 || x == -1) return -1;
14      for(--x; count_one(x) != num && x > min_int; --x);
15      if(count_one(x) == num) return x;
16      return -1;
17  }
```

count_one1 1

```
1   int count_one(int x){
2       int cnt = 0;
3       for(int i=0; i<32; ++i){
4           if(x & 1) ++cnt;
5           x >>= 1;
6       }
7       return cnt;
8   }
```

forwhile(x > 0)xx xfor

121 411

```
1   int count_one(int x){
2       x = (x & (0x55555555)) + ((x >> 1) & (0x55555555));
3       x = (x & (0x33333333)) + ((x >> 2) & (0x33333333));
4       x = (x & (0x0f0f0f0f)) + ((x >> 4) & (0x0f0f0f0f));
5       x = (x & (0x00ff00ff)) + ((x >> 8) & (0x00ff00ff));
6       x = (x & (0x0000ffff)) + ((x >> 16) & (0x0000ffff));
7       return x;
8   }
```

11011101001 011100001 11110011
1101110->1101111->1110000->1110001->1110010->1110011
() 32 3110100..001000..00 int 0100..00 0100..001 0111..1 -1-111
11100..00 1100..0111

```
int next1(int x){
    int xx = x, bit = 0;
    for(; (x&1) != 1 && bit < 32; x >>= 1, ++bit);
    for(; (x&1) != 0 && bit < 32; x >>= 1, ++bit);
    if(bit == 31) return -1; //011.., none satisify
    x |= 1;
    x <<= bit; // wtf, x<<32 != 0,so use next line to make x=0
    if(bit == 32) x = 0; // for 11100..00
    int num1 = count_one(xx) - count_one(x);
    int c = 1;
    for(; num1 > 0; x |= c, --num1, c <<= 1);
    return x;
}
```

```
int previous1(int x){
    int xx = x, bit = 0;
    for(; (x&1) != 0 && bit < 32; x >>= 1, ++bit);
    for(; (x&1) != 1 && bit < 32; x >>= 1, ++bit);
    if(bit == 31) return -1; //100..11, none satisify
    x -= 1;
    x <<= bit;
    if(bit == 32) x = 0;
    int num1 = count_one(xx) - count_one(x);
    x >>= bit;
    for(; num1 > 0; x = (x<<1) | 1, --num1, --bit);
    for(; bit > 0; x <<= 1, --bit);
    return x;
}
```

```
#include <iostream>
#include <vector>
using namespace std;

void print_binary(int x) {
    vector<int> v;
    int cnt = 0, mask = 1;
    while (cnt < 32) {
        if (x & mask) v.push_back(1);
        else v.push_back(0);
        mask <<= 1;
        ++cnt;
    }

    cnt = 0;
    while ( !v.empty() ) {
        if (cnt % 4 == 0) cout << "␣";
        if (cnt % 8 == 0) cout << "␣␣";
        cout << v.back();
        v.pop_back();
        ++cnt;
    }
```

```
23      cout << endl;
24  }
25
26  int count1(int x) {
27      int cnt = 0;
28      for (int i = 0; i < 32; ++i) {  // for is safer !!!!!!
29          if (x & 1) ++cnt;
30          x >>= 1;
31      }
32      return cnt;
33  }
34
35  int count(int x) {
36      int cnt = 0;
37      while (x > 0) {
38          if (x & 1) ++cnt;
39          x >>= 1;
40      }
41      return cnt;
42  }
43
44  int count_one (int x) {
45      x = (x & (0x55555555)) + ((x >> 1) & (0x55555555));
46      x = (x & (0x33333333)) + ((x >> 2) & (0x33333333));
47      x = (x & (0x0f0f0f0f)) + ((x >> 4) & (0x0f0f0f0f));
48      x = (x & (0x00ff00ff)) + ((x >> 8) & (0x00ff00ff));
49      x = (x & (0x0000ffff)) + ((x >> 16) & (0x0000ffff));
50      return x;
51  }
52
53  int count_oneP (int x) {
54      print_binary(x);
55      cout << endl;
56
57      cout << "1:" << endl;
58      print_binary(x);
59      print_binary(0x55555555);
60      print_binary(x & (0x55555555));
61      cout << endl;
62      print_binary(x >> 1);
63      print_binary(0x55555555);
64      print_binary((x >> 1) & (0x55555555));
65      cout << endl;
66      print_binary(x & (0x55555555));
67      print_binary((x >> 1) & (0x55555555));
68      x = (x & (0x55555555)) + ((x >> 1) & (0x55555555));
69      print_binary(x);
70      cout << endl;
71
72      cout << "2:" << endl;
73      print_binary(x);
74      print_binary(0x33333333);
75      print_binary(x & (0x33333333));
76      cout << endl;
77      print_binary(x >> 2);
78      print_binary(0x33333333);
79      print_binary((x >> 2) & (0x33333333));
80      cout << endl;
81      print_binary(x & (0x33333333));
```

```
82      print_binary((x >> 2) & (0x33333333));
83      x = (x & (0x33333333)) + ((x >> 2) & (0x33333333));
84      print_binary(x);
85      cout << endl;
86
87      cout << "4:" << endl;
88      print_binary(x);
89      print_binary(0x0f0f0f0f);
90      print_binary(x & (0x0f0f0f0f));
91      cout << endl;
92      print_binary(x >> 4);
93      print_binary(0x0f0f0f0f);
94      print_binary((x >> 4) & (0x0f0f0f0f));
95      cout << endl;
96      print_binary(x & (0x0f0f0f0f));
97      print_binary((x >> 4) & (0x0f0f0f0f));
98      x = (x & (0x0f0f0f0f)) + ((x >> 4) & (0x0f0f0f0f));
99      print_binary(x);
100     cout << endl;
101
102     cout << "8:" << endl;
103     print_binary(x);
104     print_binary(0x00ff00ff);
105     print_binary(x & (0x00ff00ff));
106     cout << endl;
107     print_binary(x >> 8);
108     print_binary(0x00ff00ff);
109     print_binary((x >> 8) & (0x00ff00ff));
110     cout << endl;
111     print_binary(x & (0x00ff00ff));
112     print_binary((x >> 8) & (0x00ff00ff));
113     x = (x & (0x00ff00ff)) + ((x >> 8) & (0x00ff00ff));
114     print_binary(x);
115     cout << endl;
116
117     cout << "16:" << endl;
118     print_binary(x);
119     print_binary(0x0000ffff);
120     print_binary(x & (0x0000ffff));
121     cout << endl;
122     print_binary(x >> 16);
123     print_binary(0x0000ffff);
124     print_binary((x >> 16) & (0x0000ffff));
125     cout << endl;
126     print_binary(x & (0x0000ffff));
127     print_binary((x >> 16) & (0x0000ffff));
128     x = (x & (0x0000ffff)) + ((x >> 16) & (0x0000ffff));
129     print_binary(x);
130     return x;
131 }
132
133 /* mine, not complete
134 int next(int x) {
135     int num = count(x);
136     int val = x+1;
137     while ( count(val) != num )
138         val += 1;
139     return val;
140 } */
```

```
141
142 int next1(int x) {
143     int max_int = ~(1<<31);
144     int num = count(x);
145     if (x == 0 || x == -1) return -1;
146     for (++x; count(x) != num && x < max_int; ++x);
147     if (count(x) == num) return x;
148     return -1;
149 }
150
151 /* mine, not complete
152 int previous(int x) {
153     int num = count(x);
154     int val = x-1;
155     while ( count(val) != num )
156         val -= 1;
157     return val;
158 } */
159
160 int previous1(int x) {
161     int min_int = (1 << 31);
162     int num = count(x);
163     if (x == 0 || x == -1) return -1;
164     for (--x; count(x) != num && x > min_int; --x);
165     if (count(x) == num) return x;
166     return -1;
167 }
168
169 int next2(int x) {
170     int xx = x, bit = 0;
171     for (; (x&1) != 1 && bit < 32; x >>= 1, ++bit);
172     for (; (x&1) != 0 && bit < 32; x >>= 1, ++bit);
173     if (bit == 31) return -1; // 011---, none satisfy
174     x |= 1;
175     x <<= bit; // wtf, x<<32 != 0, so use next line to make x=0
176     if (bit == 32) x = 0; // for 11100---00
177     int num1 = count(xx) - count(x);
178     int c = 1;
179     for (; num1 > 0; x |= c, --num1, c <<= 1);
180     return x;
181 }
182
183 int previous2(int x){
184     int xx = x, bit = 0;
185     for(; (x&1) != 0 && bit < 32; x >>= 1, ++bit);
186     for(; (x&1) != 1 && bit < 32; x >>= 1, ++bit);
187     if(bit == 31) return -1; //100..11, none satisify
188     x -= 1;
189     x <<= bit;
190     if(bit == 32) x = 0;
191     int num1 = count_one(xx) - count_one(x);
192     x >>= bit;
193     for(; num1 > 0; x = (x<<1) | 1, --num1, --bit);
194     for(; bit > 0; x <<= 1, --bit);
195     return x;
196 }
197
198 int main() {
199     //int x = (1<<30) | (1<<28) | (1<<25) | (1<<21) | (1<<19) | (1<<15)
```

```
200     //       | (1<<13) | (1<<10) | (1<<8) | (1<<6) | (1<<5) | (1<<2);
201
202     int x = -976756; // (1<<31)+(1<<29); // -8737776;
203
204     //int cnt = count_oneP(x);
205     //cout << "cnt: " << cnt << endl;
206
207     print_binary(x);
208     cout << endl;
209     print_binary( next1(x) );
210     print_binary( next2(x) );
211     cout << endl;
212     print_binary( previous1(x) );
213     print_binary( previous2(x) );
214
215     return 0;  // the result may have problem
216 }
217
218
219 /*
220 jenny@jenny-G50VT ~/docu/iv/ctci $ g++ thr.cpp
221 jenny@jenny-G50VT ~/docu/iv/ctci $ ./a.out
222     0101 0010   0010 1000   1010 0101   0110 0100
223
224 1:
225     0101 0010   0010 1000   1010 0101   0110 0100
226     0101 0101   0101 0101   0101 0101   0101 0101
227     0101 0000   0000 0000   0000 0101   0100 0100
228
229     0010 1001   0001 0100   0101 0010   1011 0010
230     0101 0101   0101 0101   0101 0101   0101 0101
231     0000 0001   0001 0100   0101 0000   0001 0000
232
233     0101 0000   0000 0000   0000 0101   0100 0100
234     0000 0001   0001 0100   0101 0000   0001 0000
235     0101 0001   0001 0100   0101 0101   0101 0100
236
237 2:
238     0101 0001   0001 0100   0101 0101   0101 0100
239     0011 0011   0011 0011   0011 0011   0011 0011
240     0001 0001   0001 0000   0001 0001   0001 0000
241
242     0001 0100   0100 0101   0001 0101   0101 0101
243     0011 0011   0011 0011   0011 0011   0011 0011
244     0001 0000   0000 0001   0001 0001   0001 0001
245
246     0001 0001   0001 0000   0001 0001   0001 0000
247     0001 0000   0000 0001   0001 0001   0001 0001
248     0010 0001   0001 0001   0010 0010   0010 0001
249
250 4:
251     0010 0001   0001 0001   0010 0010   0010 0001
252     0000 1111   0000 1111   0000 1111   0000 1111
253     0000 0001   0000 0001   0000 0010   0000 0001
254
255     0000 0010   0001 0001   0001 0010   0010 0010
256     0000 1111   0000 1111   0000 1111   0000 1111
257     0000 0010   0000 0001   0000 0010   0000 0010
258
```

```
    0000 0001   0000 0001   0000 0010   0000 0001
    0000 0010   0000 0001   0000 0010   0000 0010
    0000 0011   0000 0010   0000 0100   0000 0011

8:
    0000 0011   0000 0010   0000 0100   0000 0011
    0000 0000   1111 1111   0000 0000   1111 1111
    0000 0000   0000 0010   0000 0000   0000 0011

    0000 0000   0000 0011   0000 0010   0000 0100
    0000 0000   1111 1111   0000 0000   1111 1111
    0000 0000   0000 0011   0000 0000   0000 0100

    0000 0000   0000 0010   0000 0000   0000 0011
    0000 0000   0000 0011   0000 0000   0000 0100
    0000 0000   0000 0101   0000 0000   0000 0111

16:
    0000 0000   0000 0101   0000 0000   0000 0111
    0000 0000   0000 0000   1111 1111   1111 1111
    0000 0000   0000 0000   0000 0000   0000 0111

    0000 0000   0000 0000   0000 0000   0000 0101
    0000 0000   0000 0000   1111 1111   1111 1111
    0000 0000   0000 0000   0000 0000   0000 0101

    0000 0000   0000 0000   0000 0000   0000 0111
    0000 0000   0000 0000   0000 0000   0000 0101
    0000 0000   0000 0000   0000 0000   0000 1100
cnt: 12

*/
```

5.4 Explain what the following code does: ((n & (n-1)) == 0).

((n & (n-1)) == 0)

Solution:

2 n=002

(n > 0) && ((n & (n-1)) == 0)

5.5 Write a function to determine the number of bits required to convert integer A to integer B.

Input: 31, 14

Output: 2

AB

3114

2

Solution:

ABAB AB1

```
int count_one(int x){
    x = (x & (0x55555555)) + ((x >> 1) & (0x55555555));
    x = (x & (0x33333333)) + ((x >> 2) & (0x33333333));
    x = (x & (0x0f0f0f0f)) + ((x >> 4) & (0x0f0f0f0f));
    x = (x & (0x00ff00ff)) + ((x >> 8) & (0x00ff00ff));
    x = (x & (0x0000ffff)) + ((x >> 16) & (0x0000ffff));
    return x;
}

int convert_num(int a, int b){
```

```
11      return count_one(a^b);
12  }
```

```
1   #include <iostream>
2   using namespace std;
3
4   int count(int x) {
5       int cnt = 0;
6       for (int i = 0; i < 32; ++i) {  // refer to ch5thr.cpp
7           if (x & 1) ++cnt;
8           x >>= 1;
9       }
10      return cnt;
11  }
12
13  int count_one(int x){
14      x = (x & (0x55555555)) + ((x >> 1) & (0x55555555));
15      x = (x & (0x33333333)) + ((x >> 2) & (0x33333333));
16      x = (x & (0x0f0f0f0f)) + ((x >> 4) & (0x0f0f0f0f));
17      x = (x & (0x00ff00ff)) + ((x >> 8) & (0x00ff00ff));
18      x = (x & (0x0000ffff)) + ((x >> 16) & (0x0000ffff));
19      return x;
20  }
21
22  int convert_num(int a, int b) {
23      return count(a ^ b);
24  }
25
26  int main() {
27      int a = 31, b = 14;
28      cout << "Bits␣needs␣to␣convert:␣" << convert_num(a, b) << endl;
29      return 0;
30  }
```

5.6 Write a program to swap odd and even bits in an integer with as few instructions as possible (e.g., bit 0 and bit 1 are swapped, bit 2 and bit 3 are swapped, etc).

(0123)

Solution:

```
1   int swap_bits(int x){
2       return ((x & 0x55555555) << 1) | ((x >> 1) & 0x55555555);
3   }
```

```
1   int swap_bits1(int x){
2       return ((x & 0x55555555) << 1) | ((x & 0xAAAAAAAA) >> 1);
3   }
```

Hackers delight 1:P

```
1   #include <iostream>
2   using namespace std;
3
4   int swap_bits(int x) {
5       return ((x & 0x55555555) << 1) | ((x>>1) & 0x55555555);
```

```
}

int swap_bits1(int x) {
    return ((x & 0x55555555) << 1) | ((x & 0xaaaaaaaa) >> 1);
}

void print_binary (int x) {
    string s = "␣";
    for (int i = 0; i < 32 && x != 0; ++i, x >>= 1) {  // try me !!!
        if (i % 4 == 0) s += "␣";
        if (i % 8 == 0) s += "␣␣"; // for printing propose

        if (x & 1) s += "1";
        else s += "0";
    }
    cout << s << endl;
}

int main() {
    int x = -7665543;
    print_binary(x);
    print_binary(swap_bits(x));
    print_binary(swap_bits1(x));
    return 0;
}

/*
jenny@jenny-G50VT ~/docu/iv/ctci $ g++ six.cpp
jenny@jenny-G50VT ~/docu/iv/ctci $ ./a.out
    1001 1110   0001 0000   1101 0001   1111 1111
    0110 1101   0010 0000   1110 0010   1111 1111
    0110 1101   0010 0000   1110 0010   1111 1111
*/
```

5.7 An array A[1n] contains all the integers from 0 to n except for one number which is missing. In this problem, we cannot access an entire integer in A with a single operation. The elements of A are represented in binary, and the only operation we can use to access them is fetch the jth bit of A[i], which takes constant time. Write code to find the missing integer. Can you do it in O(n) time?

A[1n]0n A[i]iA 0/1A[i]j O(n)

Solution:

fetch(a, i, j) a[i]j a[i]

a[i]fetcha[i] a32

```
int get(int a[], int i){
    int ret = 0;
    for(int j=31; j>=0; --j)
        ret = (ret << 1) | fetch(a, i, j);
    return ret;
}
```

get(a, i)a[i]bool true

```
int missing(int a[], int n){
    bool *b = new bool[n+1];
    memset(b, false, (n+1)*sizeof(bool));
    for(int i=0; i<n; ++i)
        b[get(a, i)] = true;
    for(int i=0; i<n+1; ++i){
```

```
7           if(!b[i]) return i;
8       }
9       delete[] b;
10  }
```

ajfetch(a, j) a[i]j a[i]a[i]310(32) 32*i+3132*i

```
1   int get1(int a[], int i){
2       int ret = 0;
3       int base = 32*i;
4       for(int j=base+31; j>=base; --j)
5           ret = (ret << 1) | fetch1(a, j);
6       return ret;
7   }
```

```
1   #include <iostream>
2   #include <cstring>
3   using namespace std;
4
5   int fetch (int a[], int i, int j) {
6       return (a[i] >> j) & 1; // return 0 or 1
7   }
8
9   int get(int a[], int i) {
10      int ret = 0;
11      for (int j = 31; j >= 0; --j)
12          ret = (ret << 1) | fetch(a, i, j);
13      return ret;
14  }
15
16  int fetch1 (int a[], int i) {
17      //return (a[i/32] >> (32 - i % 32)) & 1;  // not sure what's going on
18      return (a[i/32] >> (i % 32)) & 1;
19  }
20
21  int get1(int a[], int i) {
22      int ret = 0;
23      int base = 32 * i;
24      for (int j = base + 31; j >= base; --j)
25          ret = (ret << 1) | fetch1(a, j);
26      return ret;
27  }
28
29  int missing(int a[], int n) {
30      bool *b = new bool[n+1];
31      memset(b, false, (n+1)*sizeof(bool));
32
33      for (int i = 0; i < n; ++i)
34          b[get(a, i)] = true;
35      for (int i = 0; i <= n; ++i)
36          if ( !b[i] )
37              return i;
38      delete[] b;
39  }
40
41  int missing1(int a[], int n){
42      bool *b = new bool[n+1];
43      memset(b, false, (n+1)*sizeof(bool));
44      for(int i=0; i<n; ++i)
```

```
45          b[get1(a, i)] = true;
46      for(int i=0; i<n+1; ++i){
47          if(!b[i]) return i;
48      }
49      delete[] b;
50  }
51
52  int main() {
53      int a[] = {0, 1, 2, 3, 4, 5, 7, 8, 9, 10};
54      cout << missing(a, 10) << endl;
55      cout << missing1(a, 10) << endl;
56      return 0;
57  }
```

Chapter 6

Brain Teasers

6.1 Add arithmetic operators (plus, minus, times, divide) to make the following expression true: 3 1 3 6 = 8. You can use any parentheses youd like.

6.2 There is an 8x8 chess board in which two diagonally opposite corners have been cutoff. You are given 31 dominos, and a single domino can cover exactly two squares. Can you use the 31 dominos to cover the entire board? Prove your answer (by providing an example, or showing why its impossible).

6.3 You have a five quart jug and a three quart jug, and an unlimited supply of water (but no measuring cups). How would you come up with exactly four quarts of water? NOTE: The jugs are oddly shaped, such that filling up exactly half of the jug would be impossible.

6.4 A bunch of men are on an island. A genie comes down and gathers everyone together and places a magical hat on some peoples heads (i.e., at least one person has a hat). The hat is magical: it can be seen by other people, but not by the wearer of the hat himself. To remove the hat, those (and only those who have a hat) must dunk themselves underwater at exactly midnight. If there are n people and c hats, how long does it take the men to remove the hats? The men cannot tell each other (in any way) that they have a hat.

FOLLOW UP

Prove that your solution is correct.

6.5 There is a building of 100 floors. If an egg drops from the Nth floor or above it will break. If its dropped from any floor below, it will not break. Youre given 2 eggs. Find N, while minimizing the number of drops for the worst case.

6.6 There are one hundred closed lockers in a hallway. A man begins by opening all one hundred lockers. Next, he closes every second locker. Then he goes to every third locker and closes it if it is open or opens it if it is closed (e.g., he toggles every third locker). After his one hundredth pass in the hallway, in which he toggles only locker number one hundred, how many lockers are open?

Chapter 7

Object Oriented Design

7.1 Design the data structures for a generic deck of cards. Explain how you would subclass it to implement particular card games.

7.2 Imagine you have a call center with three levels of employees: fresher, technical lead(TL), product manager (PM). There can be multiple employees, but only one TL or PM. An incoming telephone call must be allocated to a fresher who is free. If a fresher cant handle the call, he or she must escalate the call to technical lead. If the TL is not free or not able to handle it, then the call should be escalated to PM. Design the classes and data structures for this problem. Implement a method getCallHandler().

7.3 Design a musical juke box using object oriented principles.

7.4 Design a chess game using object oriented principles.

7.5 Design the data structures for an online book reader system.

7.6 Implement a jigsaw puzzle. Design the data structures and explain an algorithm to solve the puzzle.

7.7 Explain how you would design a chat server. In particular, provide details about the various backend components, classes, and methods. What would be the hardest problems to solve?

7.8 Othello is played as follows: Each Othello piece is white on one side and black on the other. When a piece is surrounded by its opponents on both the left and right sides, or both the top and bottom, it is said to be captured and its color is flipped. On your turn, you must capture at least one of your opponents pieces. The game ends when either user has no more valid moves, and the win is assigned to the person with the most pieces. Implement the object oriented design for Othello.

7.9 Explain the data structures and algorithms that you would use to design an in-memory file system. Illustrate with an example in code where possible.

7.10 Describe the data structures and algorithms that you would use to implement a garbage collector in C++.

Chapter 8

Recursion

8.1 Write a method to generate the nth Fibonacci number.

n

Solution:

```
f(1) = f(2) = 1;
f(n) = f(n-1) + f(n-2);
```

XDlong long

```
typedef long long ll;
```

```
ll fib(ll n){
    if(n < 1) return -1;
    if(n == 1 || n == 2) return 1;
    else return fib(n-1) + fib(n-2);
}
```

()

```
ll fib1(ll n){
    if(n < 1) return -1;
    if(n == 1 || n == 2) return 1;
    ll a = 1, b = 1;
    for(ll i=3; i<=n; ++i){
        ll c = a + b;
        a = b;
        b = c;
    }
    return b;
}
```

O(1)O(n)

f(1)=f(2)=1

naive(nn0)

```
ll pow(ll m, ll n){
    ll res = 1;
    for(ll i=0; i<n; ++i)
        res *= m;
    return res;
}
```

O(n)m13 1313=1101res=1.
m13 = m1 * m4 * m8
1311011 res110111resm1 0resm2 1resm4 1resm8
res = m1 * m4 * m8
res(0)

```
ll pow1(ll m, ll n){
    ll res = 1;
    while(n > 0){
        if(n&1) res *= m;
        m *= m;
        n >>= 1;
    }
    return res;
}
```

O(logn)OK

```
void pow(ll s[2][2], ll a[2][2], ll n){
    while(n > 0){
        if(n&1) mul(s, s, a);
        mul(a, a, a);
        n >>= 1;
    }
}
```

return (2*2)

```
void mul(ll c[2][2], ll a[2][2], ll b[2][2]){
    ll t[4];
    t[0] = a[0][0]*b[0][0] + a[0][1]*b[1][0];
    t[1] = a[0][0]*b[0][1] + a[0][1]*b[1][1];
    t[2] = a[1][0]*b[0][0] + a[1][1]*b[1][0];
    t[3] = a[1][0]*b[0][1] + a[1][1]*b[1][1];
    c[0][0] = t[0];
    c[0][1] = t[1];
    c[1][0] = t[2];
    c[1][1] = t[3];
}
```

nO(logn)

```
ll fib2(ll n){
    if(n < 1) return -1;
    if(n == 1 || n == 2) return 1;

    ll a[2][2] = { {1, 1}, {1, 0} };
    ll s[2][2] = { {1, 0}, {0, 1} };
    pow(s, a, n-2);
    return s[0][0] + s[0][1];
}
```

```
#include <iostream>
using namespace std;

typedef long long ll;

ll fib(ll n) {
    if (n < 1) return -1;
    if (n == 1 | n == 2) return 1;
```

```
9       else
10          return fib(n-1) + fib(n-2);
11  }
12
13  ll fibm(ll n) {     // mine, works
14      if (n < 1) return -1;
15      if (n == 1 | n == 2) return 1;
16
17      ll fib[n-1];
18      fib[0] = 1;
19      fib[1] = 1;
20      for (int i = 3; i <= n; ++i )
21          fib[i-1] = fib[i-2] + fib[i-3];
22      return fib[n-1];
23
24  }
25
26  ll fib1(ll n) {
27      if (n < 1) return -1;
28      if (n == 1 | n == 2) return 1;
29
30      ll a = 1, b = 1;
31      for (int i = 3; i <= n; ++i ) {
32          ll c = a + b;
33          a = b;
34          b = c;
35      }
36      return b;
37  }
38
39  ll pow(ll m, ll n) {
40      ll res = 1;
41      for (ll i = 0; i < n; ++i)
42          res *= m;
43      return res;
44  }
45
46  ll pow1(ll m, ll n) {
47      ll res = 1;
48      while (n > 0) {
49          if (n & 1) res *= m;
50                  m *= m;
51                  n >>= 1;
52      }
53      return res;
54  }
55
56  void mul(ll c[2][2], ll a[2][2], ll b[2][2]) {
57      ll t[4];
58      t[0] = a[0][0]*b[0][0] + a[0][1]*b[1][0];
59      t[1] = a[0][0]*b[0][1] + a[0][1]*b[1][1];
60      t[2] = a[1][0]*b[0][0] + a[1][1]*b[1][0];
61      t[3] = a[1][0]*b[0][1] + a[1][1]*b[1][1];
62      c[0][0] = t[0];
63      c[0][1] = t[1];
64      c[1][0] = t[2];
65      c[1][1] = t[3];
66  }
67
```

```
68 void pow2(ll s[2][2], ll a[2][2], ll n) {
69     while (n > 0) {
70         if (n & 1) mul(s, s, a);
71         mul(a, a, a);
72         n >>= 1;
73     }
74 }
75
76 ll fib2(ll n) {
77     if (n < 1) return -1;
78     if (n == 1 || n == 2) return 1;
79
80     ll a[2][2] = { {1, 1}, {1, 0} };
81     ll s[2][2] = { {1, 0}, {0, 1} };
82     pow2(s, a, n-2);
83     return s[0][0] + s[0][1];
84 }
85
86 int main() {
87     for (int i = 1; i < 20; ++i)
88         cout << fib1(i) << endl;
89     cout << endl;
90
91     for (int i = 1; i < 20; ++i)
92         cout << fib2(i) << endl;
93
94     return 0;
95 }
```

8.2 Imagine a robot sitting on the upper left hand corner of an NxN grid. The robot can only move in two directions: right and down. How many possible paths are there for the robot?

FOLLOW UP

Imagine certain squares are off limits, such that the robot can not step on them. Design an algorithm to get all possible paths for the robot.

N*N

Solution:

m*n(1, 1) (m, n)

(i, j)(1, 1)path(i, j)

path(i, j) = path(i-1, j) + path(i, j-1)

(i-1, j)(i, j-1) (i, j) (1, 1)

```
1 ll path(ll m, ll n){
2     if(m == 1 || n == 1) return 1;
3     else return path(m-1, n) + path(m, n-1);
4 }
```

lllong long

(1, 1)(m, n)m-1n-1 (m-1+n-1)(m-1) (n-1)

C(m-1+n-1, m-1)=(m-1+n-1)! / ((m-1)! * (n-1)!)

```
1 ll fact(ll n){
2     if(n == 0) return 1;
3     else return n*fact(n-1);
```

```
}
ll path1(ll m, ll n){
    return fact(m-1+n-1)/(fact(m-1)*fact(n-1));
}
```

(XD)

(m, n) (1, 1)

```
bool get_path(int m, int n){
    point p; p.x=n; p.y=m;
    sp.push(p);
    if(n==1 && m==1) return true;
    bool suc = false;
    if(m>1 && g[m-1][n])
        suc = get_path(m-1, n);
    if(!suc && n>1 && g[m][n-1])
        suc = get_path(m, n-1);
    if(!suc) sp.pop();
    return suc;
}
```

gM*N10 (1, 1) (M, N)

```
void print_paths(int m, int n, int M, int N, int len){
    if(g[m][n] == 0) return;
    point p; p.x=n; p.y=m;
    vp[len++] = p;
    if(m == M && n == N){
        for(int i=0; i<len; ++i)
            cout<<"("<<vp[i].y<<",␣"<<vp[i].x<<")"<<"␣";
        cout<<endl;
    }
    else{
        print_paths(m, n+1, M, N, len);
        print_paths(m+1, n, M, N, len);
    }
}
```

8.2.in

```
3 4
1 1 1 0
0 1 1 1
1 1 1 1
```

```
one of the paths:
(1, 1) (1, 2) (1, 3) (2, 3) (2, 4) (3, 4)
all paths:
(1, 1) (1, 2) (1, 3) (2, 3) (2, 4) (3, 4)
(1, 1) (1, 2) (1, 3) (2, 3) (3, 3) (3, 4)
(1, 1) (1, 2) (2, 2) (2, 3) (2, 4) (3, 4)
(1, 1) (1, 2) (2, 2) (2, 3) (3, 3) (3, 4)
(1, 1) (1, 2) (2, 2) (3, 2) (3, 3) (3, 4)
```

```
#include <iostream>
#include <stack>
#include <cstdio>
using namespace std;

```

```
6  typedef long long ll;
7
8  struct point {
9      int x;
10     int y;
11 };
12 stack<point> sp;
13 const int MAXN = 20;
14 int g[MAXN][MAXN];
15 point vp[MAXN+MAXN];
16
17 ll path(ll m, ll n) {
18     if (m == 1 || n == 1) return 1;
19     else return path(m-1, n) + path(m, n-1);
20 }
21
22 ll fact(ll n) {
23     if (n == 0) return 1;
24     else return n*fact(n-1);
25 }
26
27 ll path1(ll m, ll n) {
28     return fact(m-1+n-1)/(fact(m-1)*fact(n-1));
29 }
30
31 bool get_path(int m, int n) {
32     point p; p.x = m; p.y = n;
33     sp.push(p);
34     if (m == 1 && n == 1) return true;
35
36     bool suc = false;
37     if (m > 1 && g[m-1][n])
38         suc = get_path(m-1, n);
39     if (!suc && n>1 && g[m][n-1])
40         suc = get_path(m, n-1);
41     if (!suc) sp.pop();
42     return suc;
43 }
44
45 void print_paths(int m, int n, int M, int N, int len) {
46     if (g[m][n] == 0) return;
47     point p; p.x = m; p.y = n;
48     vp[len++] = p;
49     if (m == M && n == N) {
50         for (int i = 0; i < len; ++i)
51             cout << "(" << vp[i].x << ",␣" << vp[i].y << ")" << "␣";
52         cout << endl;
53     } else {
54         print_paths(m+1, n, M, N, len);
55         print_paths(m, n+1, M, N, len);
56     }
57 }
58
59 int main() {
60     freopen("ch88.2.in", "r", stdin);
61
62     for (int i = 1; i < 10; ++i)
63         cout << path(i, i) << endl;
64     cout << endl;
```

```
65
66      for (int i = 1; i < 10; ++i)
67          cout << path1(i, i) << endl;
68      cout << endl;
69
70      int M, N;
71      cin >> M >> N;
72      for (int i = 1; i <= M; ++i)
73          for (int j = 1; j <= N; ++j)
74              cin >> g[i][j];
75      cout << "one of the paths: " << endl;
76      get_path(M, N);
77
78      while( !sp.empty() ) {
79          point p = sp.top();
80          cout << "(" << p.x << ", " << p.y << ")" << " ";
81          sp.pop();
82      }
83      cout << endl << "all paths: " << endl;
84      print_paths(1, 1, M, N, 0);
85
86      fclose(stdin);
87      return 0;
88  }
```

8.3 Write a method that returns all subsets of a set.

Solution:
2n () 1, 2, 30, 0, 0 1, 32(0)13(1) 1, 0, 11, 2, 3000111 8n02n -1 2n 1 OK

```
1  typedef vector<vector<int> > vvi;
2  typedef vector<int> vi;
3  vvi get_subsets(int a[], int n){ //O(n2^n)
4      vvi subsets;
5      int max = 1<<n;
6      for(int i=0; i<max; ++i){
7          vi subset;
8          int idx = 0;
9          int j = i;
10         while(j > 0){
11             if(j&1){
12                 subset.push_back(a[idx]);
13             }
14             j >>= 1;
15             ++idx;
16         }
17         subsets.push_back(subset);
18     }
19     return subsets;
20 }
```

1, 2, 3 12, 32, 31, 2, 3 2, 31 1, 2, 32, 3 1, 2, 32, 3

```
1  vvi get_subsets1(int a[], int idx, int n){
2      vvi subsets;
3      if(idx == n){
4          vi subset;
5          subsets.push_back(subset); //empty set
6      }
7      else{
```

```
8           vvi rsubsets = get_subsets1(a, idx+1, n);
9           int v = a[idx];
10          for(int i=0; i<rsubsets.size(); ++i){
11              vi subset = rsubsets[i];
12              subsets.push_back(subset);
13              subset.push_back(v);
14              subsets.push_back(subset);
15          }
16      }
17      return subsets;
18  }
```

```
1   #include <iostream>
2   #include <vector>
3   using namespace std;
4
5   typedef vector< vector<int> > vvi;
6   typedef vector<int>  vi;
7
8   vvi get_subsets(int a[], int n) {  // O(n*2^n)
9       vvi subsets;
10      int max = 1 << n;
11      for (int i = 0; i < max; ++i) {
12          vi subset;
13          int idx = 0;
14          int j = i;
15          while (j > 0) {
16           // if (j & 1) subset.push_back(a[idx++]); WRONG, idx++, considered only one condition
17              if (j & 1)
18                  subset.push_back(a[idx]);
19              ++idx;          // attention
20              j >>= 1;
21          }
22          subsets.push_back(subset);
23      }
24      return subsets;
25  }
26
27  vvi get_subsets1(int a[], int idx, int n) {  // not understanding, difficult !!!
28      vvi subsets;
29      if (idx == n) {
30          vi subset;
31          subsets.push_back(subset);  // empty set
32      } else {
33          vvi rsubsets = get_subsets1(a, idx+1, n);
34          int v = a[idx];
35          for (int i = 0; i < rsubsets.size(); ++i) {
36              vi subset = rsubsets.at(i);
37              subsets.push_back(subset);
38              subset.push_back(v);
39              subsets.push_back(subset);
40          }
41      }
42      return subsets;
43  }
44
45  void print_subsets(vvi subsets) {
46      cout << "Subsets: " << endl;
```

```
47      for (int i = 0; i < subsets.size(); ++i) {
48          vi subset = subsets.at(i);
49          cout << "Subset Vector #" << i << ": ";
50          for (int j = 0; j < subset.size(); ++j)
51           // cout << subset.at(j) << " ";
52              cout << subset[j] << " ";     // both works just fine!!!
53          cout << endl;
54      }
55      cout << "End of Subsets!" << endl;
56  }
57
58  int main() {
59      int a[] = {1, 2, 3, 4};
60      vvi sub = get_subsets(a, 4);
61      vvi sub1 = get_subsets1(a, 0, 4);
62      print_subsets(sub);
63      print_subsets(sub1);
64      return 0;
65  }
```

8.4 Write a method to compute all permutations of a string

Solution:
nA(n, n)=n! "abc" permu

abc0apermubc bc, cba() abc(3)abc,bac,bca 0

```
1   typedef vector<string> vs;
2
3   vs permu(string s){
4       vs result;
5       if(s == ""){
6           result.push_back("");
7           return result;
8       }
9       string c = s.substr(0, 1);
10      vs res = permu(s.substr(1));
11      for(int i=0; i<res.size(); ++i){
12          string t = res[i];
13          for(int j=0; j<=t.length(); ++j){
14              string u = t;
15              u.insert(j, c);
16              result.push_back(u);
17          }
18      }
19      return result; //resultcopy()
20  }
```

abc permu abc,cba abc,acbbac,cab bac,bcaabc

```
1   vs permu1(string s){
2       vs result;
3       if(s == ""){
4           result.push_back("");
5           return result;
6       }
7       for(int i=0; i<s.length(); ++i){
8           string c = s.substr(i, 1);
9           string t = s;
```

```
10          vs res = permu1(t.erase(i, 1));
11          for(int j=0; j<res.size(); ++j){
12              result.push_back(c + res[j]);
13          }
14      }
15      return result;
16  }
```

```
1   #include <iostream>
2   #include <vector>
3   using namespace std;
4
5   typedef vector<string> vs;
6
7   void print_vs(vs result) {
8       cout << "Permutation result size: " << result.size() << endl;
9
10      for (int i = 0; i < result.size(); ++i)
11          cout << result[i] << " ";
12      cout << endl;
13  }
14
15  vs permu(string s) {
16      vs result;
17      if (s == "") {
18          result.push_back(s);
19          return result;
20      }
21
22      string c = s.substr(0, 1);
23      vs res = permu( s.substr(1) );
24      for (int i = 0; i < res.size(); ++i) {
25          string t = res[i];
26          //result.push_back(t);  // made mistake here, NOT necessary
27          for (int j = 0; j <= t.length(); ++j) {   // one >= length()
28              // string u = t.substr(0,j) + c + t.substr(j);  // equivalent method
29              string u = t;
30              u.insert(j, c);
31              result.push_back(u);
32          }
33      }
34      return result;
35  }
36
37  vs permu1(string s) {
38      vs result;
39      if (s == "") {
40          result.push_back(s);
41          return result;
42      }
43
44      for (int i = 0; i < s.length(); ++i) {
45          string c = s.substr(i,1);  // made mistake s.substr(i, i+1)
46          string rest = s.substr(0,i)+s.substr(i+1);
47       // vs res = permu1( s.substr(0,i)+s.substr(i+1) );  // equivalent
48          string t = s;
49          vs res = permu1( t.erase(i,1) );
50          for (int j = 0; j < res.size(); ++j)
```

```
51                 result.push_back(c+res[j]);
52         }
53         return result;
54     }
55
56
57     int main() {
58         string s = "abcd";
59         vs result = permu(s);
60         print_vs(result);
61         vs result1 = permu1(s);
62         print_vs(result1);
63         return 0;
64     }
```

8.5 Implement an algorithm to print all valid (e.g., properly opened and closed) combinations of n-pairs of parentheses.
EXAMPLE:
input: 3 (e.g., 3 pairs of parentheses)
output: ((())), (()()), (())(), ()(()), ()()()
n
3 3
((())), (()()), (())(), ()(()), ()()()
Solution:
)((()()) leftright (()())

```
1  (, left = 1, right = 0
2  ((, left = 2, right = 0
3  (), left = 2, right = 1
4  (()(, left = 3, right = 1
5  (()(), left = 3, right = 2
6  (()()), left = 3, right = 3
```

0

```
1  void print_pare(int l, int r, char str[], int cnt){
2      if(l<0 || r<l) return;
3      if(l==0 && r==0){
4          for(int i=0; i<cnt; ++i){
5              cout<<str[i];
6          }
7          cout<<",␣";
8      }
9      else{
10         if(l > 0){
11             str[cnt] = '(';
12             print_pare(l-1, r, str, cnt+1);
13         }
14         if(r > l){
15             str[cnt] = ')';
16             print_pare(l, r-1, str, cnt+1);
17         }
18     }
19 }
```

```
1  #include <iostream>
2  using namespace std;
3
4  void print_pare(int left, int right, char str[], int cnt) {
```

```
5       if (left < 0 || right < left) return;
6       if (left == 0 && right == 0) {
7           for (int i = 0; i < cnt; ++i)
8               cout << str[i];
9           cout << ", ";
10      } else {
11          if (left > 0) {
12              str[cnt] = '(';
13             print_pare(left-1, right, str, cnt+1);
14          }
15          if (right > left) {
16              str[cnt] = ')';
17              print_pare(left, right-1, str, cnt+1);
18          }
19      }
20  }
21
22  int main() {
23      int cnt = 3;
24      char str[2*cnt];
25      print_pare(cnt, cnt, str, 0);
26      return 0;
27  }
```

8.6 Implement the paint fill function that one might see on many image editing programs. That is, given a screen (represented by a 2-dimensional array of Colors), a point, and a new color, fill in the surrounding area until you hit a border of that color.

() ()

Solution:

() ()

```
1   bool paint_fill(color **screen, int m, int n, int x, int y, color c){
2       if(x<0 || x>=m || y<0 || y>=n) return false;
3       if(screen[x][y] == c) return false;
4       else{
5           screen[x][y] = c;
6           paint_fill(screen, m, n, x-1, y, c);
7           paint_fill(screen, m, n, x+1, y, c);
8           paint_fill(screen, m, n, x, y-1, c);
9           paint_fill(screen, m, n, x, y+1, c);
10      }
11      return true;
12  }
```

BFS ()

```
1   #include <iostream>
2   #include <cstdio>
3   using namespace std;
4
5   enum color {red, yellow, blue, green};
6
7   bool paint_fill(color **screen, int m, int n, int x, int y, color c) {
8       if (x < 0 || x >= m || y < 0 || y >= n) return false;
9
10      if (screen[x][y] == c) return false;
11      else {
12          screen[x][y] = c;
13          paint_fill(screen, m, n, x-1, y, c);
14          paint_fill(screen, m, n, x+1, y, c);
```

```
15          paint_fill(screen, m, n, x, y-1, c);
16          paint_fill(screen, m, n, x, y+1, c);
17      }
18      return true;
19  }
20
21  int main() {
22      freopen("ch88.6.in", "r", stdin);
23      int m, n;
24      cin >> m >> n;
25      color **screen = new color*[m];
26      for (int i = 0; i < m; ++i)
27          screen[i] = new color[n];
28
29      for (int i = 0; i < m; ++i)
30          for (int j = 0; j < n; ++j) {
31              int t;
32              cin >> t;
33              screen[i][j] = (color)t;
34          }
35      paint_fill(screen, 5, 5, 1, 2, green);
36
37      for (int i = 0; i < 5; ++i) {
38          for (int j = 0; j < 5; ++j)
39              cout << screen[i][j] << "␣";
40          cout << endl;
41      }
42      fclose(stdin);
43      return 0;
44  }
```

8.7 Given an infinite number of quarters (25 cents), dimes (10 cents), nickels (5 cents) and pennies (1 cent), write code to calculate the number of ways of representing n cents.

251051n

Solution:

n1

```
1  int cnt = 0;
2  void sumn(int sum, int n){
3      if(sum >= n){
4          if(sum == n) ++cnt;
5          return;
6      }
7      else{
8          sumn(sum+25, n);
9          sumn(sum+10, n);
10         sumn(sum+5, n);
11         sumn(sum+1, n);
12     }
13 }
```

1,55,1 1,55,1 5,1 2525,10,5,1 1010,5,1

```
1  int cnt = 0;
2  void sumN(int sum, int c, int n){
3      if(sum >= n){
4          if(sum == n) ++cnt;
5          return;
6      }
7      else{
```

```
8           if(c >= 25)
9               sumN(sum+25, 25, n);
10          if(c >= 10)
11              sumN(sum+10, 10, n);
12          if(c >= 5)
13              sumN(sum+5, 5, n);
14          if(c >= 1)
15              sumN(sum+1, 1, n);
16      }
17  }
```

```
1   int sum_n(int sum, int c, int n){
2       int ways = 0;
3       if(sum <= n){
4           if(sum == n) return 1;
5           if(c >= 25)
6               ways += sum_n(sum+25, 25, n);
7           if(c >= 10)
8               ways += sum_n(sum+10, 10, n);
9           if(c >= 5)
10              ways += sum_n(sum+5, 5, n);
11          if(c >= 1)
12              ways += sum_n(sum+1, 1, n);
13      }
14      return ways;
15  }
```

CTCI

```
1   int make_change(int n, int denom){
2       int next_denom = 0;
3       switch(denom){
4       case 25:
5           next_denom = 10;
6           break;
7       case 10:
8           next_denom = 5;
9           break;
10      case 5:
11          next_denom = 1;
12          break;
13      case 1:
14          return 1;
15      }
16      int ways = 0;
17      for(int i=0; i*denom<=n; ++i)
18          ways += make_change(n-i*denom, next_denom);
19      return ways;
20  }
```

i(in) 10025 025100125100-25 =75225100-25*2=50 2510 251051100 0

```
1   #include <iostream>
2   using namespace std;
3
4   int cnt = 0;
5   void sumn(int sum, int c, int n) {
6       if (sum >= n) {
7           if (sum == n) ++cnt;
```

```
8           return;
9       } else {
10          if (c >= 25) sumn(sum+25, 25, n);
11          if (c >= 10) sumn(sum+10, 10, n);
12          if (c >= 5) sumn(sum+5, 5, n);
13          if (c >= 1) sumn(sum+1, 1, n);
14      }
15  }
16
17  int sum_n(int sum, int c, int n) {
18      int ways = 0;
19      if (sum <= n) {
20          if (sum == n) return 1;
21          if (c >= 25) ways += sum_n(sum+25, 25, n);
22          if (c >= 10) ways += sum_n(sum+10, 10, n);
23          if (c >=  5) ways += sum_n(sum+ 5,  5, n);
24          if (c >=  1) ways += sum_n(sum+ 1,  1, n);
25      }
26      return ways;
27  }
28
29  int make_change(int n, int denom) {
30      int next_denom = 0;
31      switch (denom) {
32      case 25:
33          next_denom = 10;
34          break;
35      case 10:
36          next_denom = 5;
37          break;
38      case 5:
39          next_denom = 1;
40          break;
41      case 1:
42          return 1;
43      }
44      int ways = 0;
45      for (int i = 0; i*denom <= n; ++i)
46          ways += make_change(n-i*denom, next_denom);
47      return ways;
48  }
49
50  int main() {
51      int n = 10;
52      sumn(0, 25, n);
53      cout << "cnt: " << cnt << endl;
54
55      cout << make_change(n, 25) << endl;
56      cout << sum_n(0, 25, n) << endl;
57      return 0;
58  }
```

8.8 Write an algorithm to print all ways of arranging eight queens on a chess board so that none of them share the same row, column or diagonal.

8*888(2)

Solution:

8264 11 2 2233 18 8

c[i]=j ijrc[r] 8c[r]0127 c[r]==c[j]; r-j==c[r]-c[j]; r-j==c[j]-c[r]

```
#include <iostream>
using namespace std;

int c[20], n = 8, cnt = 0;

void print() {
    for (int i = 0; i < 8; ++i) {
        for (int j = 0; j < 8; ++j) {
            if (j == c[i]) cout << "1 ";
            else cout << "0 ";
        }
        cout << endl;
    }
    cout << endl;
}

void search(int r) {
    if (r == n) {
        print();
        ++cnt;
        return;
    }
    for (int i = 0; i < n; ++i) {
        c[r] = i;
        int ok = 1;
        for (int j = 0; j < r; ++j) {
            if ( c[j] == i || (r-j == c[r]-c[j]) || (r-j == c[j]-c[r]) ) {
                ok = 0;
                break;
            }
        }
        if (ok) search(r+1);
    }
}

int main() {
    search(0);
    cout << cnt << endl;
    return 0;
}
```

Chapter 9

Sorting and Searching

9.1 You are given two sorted arrays, A and B, and A has a large enough buffer at the end to hold B. Write a method to merge B into A in sorted order.

AB()AAB BA

Solution:

ABC ABCCA O(n)A A

ABA

ABA A

```
1  void merge(int a[], int b[], int n, int m){
2      int k = n + m - 1;
3      int i = n - 1, j = m - 1;
4      while(i>=0 && j>=0){
5          if(a[i] > b[j]) a[k--] = a[i--];
6          else a[k--] = b[j--];
7      }
8      while(j>=0) a[k--] = b[j--];
9  }
```

BA AAwhileB AA ABnmO(n+m)

AA

A 2(1) 12

```
1  #include <iostream>
2  using namespace std;
3
4  void merge(int a[], int b[], int n, int m) {   // O(m+n)
5      int k = m + n -1;
6      int i = n-1, j = m-1;
7
8      while (i >= 0 && j >= 0) {
9          if (a[i] > b[j]) a[k--] = a[i--];
10         else a[k--] = b[j--];
11     }
12     while (j >= 0) a[k--] = b[j--];
13 }
14
15 void swap(int &a, int &b) {
16     a = a^b;
17     b = a^b;
18     a = a^b;
19 }
```

```
20
21 void merge(int a[], int begin, int mid, int end) {
22     for (int i = begin; i <= mid; ++i) {
23         if (a[i] > a[mid+1]) {
24             swap(a[i], a[mid+1]);
25
26             for (int j = mid+1; j < end; ++j) {
27                 if (a[j] <= a[j+1]) break;
28                 swap(a[j], a[j+1]);
29             }
30         }
31     }
32 }
33
34 int main() {
35     int a[15] = {1, 5, 6, 7};
36     int b[] = {2, 3, 4, 8, 10, 12, 14};
37     int n = 4, m = 7;
38     merge(a, b, 4, 7);
39     for (int i = 0; i < m+n; ++i)
40         cout << a[i] << "␣";
41     cout << endl;
42
43     int a1[10] = {8, 9, 11, 15, 17, 1, 3, 5, 12, 18};
44     merge(a1, 0, 4, 9);
45     for (int i = 0; i < 10; ++i)
46         cout << a1[i] << "␣";
47     cout << endl;
48     return 0;
49 }
```

9.2 Write a method to sort an array of strings so that all the anagrams are next to each other.

Solution:
liveevilOK STLsort sort nAsort
sort(A, A+n);
Ppersonperson agecmp

```
1 bool cmp(person p1, person p2){
2     return p1.age < p2.age;
3 }
```

sort
sort(P, P+n, cmp);
OK sort

```
1 #include <iostream>
2 #include <algorithm>
3 using namespace std;
4
5 bool cmp(string s1, string s2) {
6     sort(&s1[0], &s1[0]+s1.length());
7     sort(&s2[0], &s2[0]+s2.length());
8     return s1 < s2;
9 }
10
11 int main() {
12     string s[] = {"abc", "cba", "bca", "xyz", "hijklmn", "fg", "gf"};
```

```
13      sort(s, s+7, cmp);
14      for (int i = 0; i < 7; ++i)
15          cout << s[i] << endl;
16      return 0;
17  }
```

9.3 Given a sorted array of n integers that has been rotated an unknown number of times, give an O(log n) algorithm that finds an element in the array. You may assume that the array was originally sorted in increasing order.

Input: find 5 in array (15 16 19 20 25 1 3 4 5 7 10 14)

Output: 8 (the index of 5 in the array)

n() O(log n)

(15 16 19 20 25 1 3 4 5 7 10 14)5

85

Solution:

O(log n) ()

int search(int a[], int low, int high, int x);

alowhighxlow <= high x

```
1  int mid = low + (high - low)/2;
2  if(a[mid] == x) return mid;
```

a[mid]x a[mid] (a[mid]>=a[low])xa[low]a[mid] high = mid - 1low = mid + 1a[mid] (a[mid]<a[low])xa[mid]a[low] low = mid + 1high = mid - 1

```
1  #include <iostream>
2  using namespace std;
3
4  int find(int a[], int low, int high, int val) {
5      if (low <= high) {
6          int mid = (low + high) >> 1;  // / 2
7          if (a[mid] == val) return mid;
8          find(a, low, mid-1, val);
9          find(a, mid+1, high, val);
10     }
11 }
12
13 int search(int a[], int low, int high, int x) {
14     while (low <= high) {
15         int mid = low + (high - low) / 2;  // different from >> 1
16         if (a[mid] == x) return mid;
17
18         if (a[mid] >= a[low]) {
19             if (a[low] <= x && x < a[mid])
20                 high = mid -1;
21             else
22                 low = mid + 1;
23         } else {
24             if (x > a[mid] && x < a[low])
25                 low = mid + 1;
26             else
27                 high = mid -1;
28         }
29     }
30     return -1;
31 }
32
33 int main() {
```

```
34      int a[] = {15, 16, 19, 20, 25, 1, 3, 4, 5, 7, 10, 14};
35      cout << find(a, 0, 11, 5) << endl;
36      cout << search(a, 0, 11, 5) << endl;
37      return 0;
38  }
```

b search-1

9.4 If you have a 2 GB file with one string per line, which sorting algorithm would you use to sort the file and why?

2GB

Solution:

2GB

X MB

1. KX*K=2GB O(nlog n)
2.
3. K

3N

9.5 Given a sorted array of strings which is interspersed with empty strings, write a method to find the location of a given string.

Example: find ball in [at, , , , ball, , , car, ,, dad, ,] will return 4

Example: find ballcar in [at, , , , , ball, car, , , dad, ,] will return -1

[at, , , , ball, , , car, ,, dad, ,] "ball"4.

[at, , , , , ball, car, , , dad, ,] "ballcar"-1.

Solution:

highhigh [low, mid-1]highx

```
1  #include <iostream>
2  using namespace std;
3
4  int search(string s[], int low, int high, string x) {
5      if (x == "") return -1;
6      while (low <= high) {
7          int mid = (low + high) >> 1;
8          int t = mid;
9          while ( s[t] == "" && t <= high)  ++t;
10         if (t > high) high = mid -1;
11         else {
12             if (s[t] == x) return t;
13             else if (s[t] < x) low = t + 1;
14             else
15                 high = mid -1;
16         }
17     }
18     return -1;
19 }
20
21 int main() {
22     string s[] = {"at", "", "", "", "ball", "", "", "car", "", "", "dad", "", ""};
23     cout << search(s, 0, 12, "ball") << endl;
24     return 0;
25 }
```

9.6 Given a matrix in which each row and each column is sorted, write a method to find an element in it.

Solution:

9.6.in()5 55*5

```
5 5
1 2 3 4 5
3 7 8 9 11
5 9 10 17 18
7 12 15 19 23
9 13 16 20 25
```

xx5 O(m+n)

```
#include <iostream>
#include <cstdio>
using namespace std;

int d[20][20];
int search(int m, int n, int val) {
    int r = 0, c = n-1;
    while (r < m && c >= 0) {
        if (d[r][c] == val) return r*n+c;
        else if (d[r][c] < val) ++r;
        else --c;
    }
    return -1;
}

int main() {
    freopen("ch99.6.in", "r", stdin);
    int m, n;
    cin >> m >> n;
    for (int i = 0; i < m; ++i)
        for (int j = 0; j < n; ++j)
            cin >> d[i][j];

    int k = search(m, n, 13);
    if (k == -1) cout << "Not found!!!" << endl;
    else cout << "Position: " << k/n << " " << k%n << endl;
    fclose(stdin);
    return 0;
}
```

9.7 A circus is designing a tower routine consisting of people standing atop one anothers shoulders. For practical and aesthetic reasons, each person must be both shorter and lighter than the person below him or her. Given the heights and weights of each person in the circus, write a method to compute the largest possible number of people in such a tower.

Input (ht, wt): (65, 100) (70, 150) (56, 90) (75, 190) (60, 95) (68, 110)

Output: The longest tower is length 6 and includes from top to bottom: (56, 90) (60,95) (65,100) (68,110) (70,150) (75,190)

( )(65, 100) (70, 150) (56, 90) (75, 190) (60, 95) (68, 110)

6 (56, 90) (60,95) (65,100) (68,110) (70,150) (75,190)

Solution:

( )

```
const int maxn = 100;
struct person{
    int h, w;
};
person p[maxn];
```

STLsort sort

```
bool cmp(person p1, person p2){
    if(p1.h == p2.h) return p1.w < p2.w;
    else return p1.h < p2.h;
}
```

sort

sort(p, p+n, cmp);

(LIS)LIS

```
int lis(person p[], int n){
    int k = 1;
    d[0] = p[0].w;
    for(int i=1; i<n; ++i){
        if(p[i].w >= d[k-1]) d[k++] = p[i].w;
        else{
            int j;
            for(j=k-1; j>=0 && d[j]>p[i].w; --j);//O(nlogn)
            d[j+1] = p[i].w;
        }
    }
    return k;
}
```

```
#include <iostream>
#include <cstdio>
#include <algorithm>
using namespace std;

const int maxn = 100;
struct person {
    int h;
    int w;
};
person p[maxn];
int d[maxn];

bool cmp(person p1, person p2) {
    if (p1.h == p2.h) return p1.w < p2.w;
    else return p1.h < p2.h;
}

int lis(person p[], int n) {
    d[0] = p[0].w;
    int k = 1;
    for (int i = 1; i < n; ++i) {
        if (p[i].w >= d[k-1])
            d[k++] = p[i].w;
        else { // binary to get O(nlogn)  NOT understanding this partx
            int j;
            for (j = k-1; j >= 0 && d[j] > p[i].w; --j);
            d[j+1] = p[i].w;
```

```
29          }
30      }
31      return k;
32  }
33
34  int main() {
35      freopen("ch99.7.in", "r", stdin);
36      int n;
37      cin >> n;
38      for (int i = 0; i < n; ++i)
39          cin >> p[i].h >> p[i].w;
40      sort(p, p+n, cmp);
41      cout << lis(p, n) << endl;
42      fclose(stdin);
43      return 0;
44  }
```

Chapter 10

Mathematical

10.1 You have a basketball hoop and someone says that you can play 1 of 2 games.
Game #1: You get one shot to make the hoop.
Game #2: You get three shots and you have to make 2 of 3 shots.
If p is the probability of making a particular shot, for which values of p should you pick one game or the other?

1
22
pp1p2
Solution:
p1p22
C(2,3)p$\hat{}$2(1-p) + p$\hat{}$3 = 3p$\hat{}$2 - 2p$\hat{}$3

p > 3p$\hat{}$2 - 2p$\hat{}$3 –> p < 0.5
1p > 0.52p = 0.5

10.2 There are three ants on different vertices of a triangle. What is the probability of collision (between any two or all of them) if they start walking on the sides of the triangle?Similarly find the probability of collision with n ants on an n vertex polygon.
33 nnn
Solution:
() 12
p = 1 - 2 * (1/2)$\hat{}$3 = 3/4
nn
p = 1 - 2 * (1/2)$\hat{}$n = 1 - 1/2$\hat{}$(n-1)

10.3 Given two lines on a Cartesian plane, determine whether the two lines would intersect.

Solution:
()x y epsilon(0.000001)

10.4 Write a method to implement *, - , / operations. You should use only the + operator.
*, - , /+
Solution:

a*b(a>0, b>0)abba ab a<baba>bba
a-ba+(-b)
a/bb0.b0( )abab 1abab

```
#include <iostream>
using namespace std;

const int INF = ~(1 << 31);

void swap(int &a, int &b) {
    a = a^b;
    b = a^b;
    a = a^b;
}

int flipsign(int a) {   // good !
    int d = a < 0 ? 1 : -1;
    int opa = 0;
    while (a != 0) {
        a += d;
        opa += d;
    }
    return opa;
}

int abs(int a) {
    if (a < 0) a = flipsign(a);
    return a;
}

bool opsign(int a, int b) {
    return (a > 0 && b < 0) || (a < 0 && b > 0);
}

int times(int a, int b) {
    if (a == 0 || b == 0) return 0;

    int aa = abs(a);
    int bb = abs(b);
    int res = 0;

    if (aa < bb) swap(aa, bb);
    for (int i = 0; i < bb; ++i)
        res += aa;

    if (opsign(a, b)) return flipsign(res);
    else return  res;
}

int minuss(int a, int b) {
    return a + flipsign(b);
}

int divide(int a, int b) {
    if (b == 0) return INF;
    int aa = abs(a), bb = abs(b);
    int res = 0;
    for (; (aa += flipsign(bb)) >= 0; ++res);
    /*
    while (aa >= bb) {
            ++res;
            aa += flipsign(bb);
    }
```

```
60      */
61
62      if (opsign(a, b)) res = flipsign(res);
63      return res;
64  }
65
66  int main() {
67      int a[] = {8, 0, -8, -5, 9};
68      int b[] = {3, 5, 3, 0, -3};
69      for (int i = 0; i < 5; ++i) {
70          cout << times(a[i], b[i]) << "␣" << minuss(a[i], b[i]) << "␣" ;
71          cout << divide(a[i], b[i]) << endl;
72      }
73      return 0;
74  }
```

10.5 Given two squares on a two dimensional plane, find a line that would cut these two squares in half.

Solution:
()

10.6 Given a two dimensional graph with points on it, find a line which passes the most number of points.

Solution:
Goolge hash map OK hash map
CTCIhashCodeObject equals

```
1  @Override
2  public int hashCode()  {
3      int sl = (int)(slope * 1000);
4      int in = (int)(intercept * 1000);
5      return sl | in;
6  }
7
8  @Override
9  public boolean equals(Object o) {
10     Line l = (Line) o;
11     if (isEqual(l.slope, slope) && isEqual(l.intercept, intercept)
12         && (infinite_slope == l.infinite_slope)) {
13         return true;
14     }
15     return false;
16 }
```

c++STLhash mapmaphashcode epsilon

```
1  #include <iostream>
2  #include <map>
3  #include <cmath>
4  #include <cstdlib>
5  #include <ctime>
6  using namespace std;
7
8  struct point {
9      double x;
10     double y;
11 };
12
13 class line {
14 private:
```

```
15      double epsilon, slope, intercept;
16      bool bslope;
17  public:
18      line () {}
19      line (point p, point q) {
20          epsilon = 0.0001;
21          if ( abs(p.x - q.x) > epsilon ) {
22              slope = (p.y - q.y) / (p.x - q.x);
23              intercept = p.y - slope * p.x;
24              bslope = true;
25          } else {
26              bslope = false;
27              intercept = p.x;
28          }
29
30      }
31      int hashcode() {
32          int sl = (int)(slope * 1000);
33          int in = (int)(intercept * 1000);
34          return sl * 1000 + in;
35      }
36      void print() {
37          cout << "y = " << slope << "x + " << intercept << endl;
38      }
39  };
40
41  line find_best_line(point *p, int point_num) {
42      line bestline;
43      bool first = true;
44      map<int, int> mii;
45      for (int i = 0; i < point_num; ++i) {
46          for (int j = 1; j < point_num; ++j) {
47              line l(p[i], p[j]);
48              if ( mii.find(l.hashcode()) == mii.end() ) {
49                  mii[l.hashcode()] = 0;
50              }
51              mii[l.hashcode()] = mii[l.hashcode()] + 1;    // Not understanding
52              if (first) {
53                  bestline = l;
54                  first = false;
55              } else {
56                  if ( mii[l.hashcode()] > mii[bestline.hashcode()] )
57                      bestline = l;
58              }
59          }
60      }
61      return bestline;
62  }
63
64  int main() {
65      srand( (unsigned)time(0) );
66      int graph_size = 100;
67      int point_num = 500;
68      point *p = new point[point_num];
69      for (int i = 0; i < point_num; ++i) {
70          p[i].x = rand()/double(RAND_MAX) * graph_size;
71          p[i].y = rand()/double(RAND_MAX) * graph_size;
72      }
73      line l = find_best_line(p, point_num);
```

```
74      l.print();
75      return 0;
76  }
```

10.7 Design an algorithm to find the kth number such that the only prime factors are 3, 5, and 7.
357k
Solution:

1	-	3_0 * 5_0 * 7_0
3	3	3_1 * 5_0 * 7_0
5	5	3_0 * 5_1 * 7_0
7	7	3_0 * 5_0 * 7_1
9	3*3	3_2 * 5_0 * 7_0
15	3*5	3_1 * 5_1 * 7_0
21	3*7	3_1 * 5_0 * 7_1
25	5*5	3_0 * 5_2 * 7_0
27	3*9	3_3 * 5_0 * 7_0
35	5*7	3_0 * 5_1 * 7_1
45	5*9	3_2 * 5_1 * 7_0
49	7*7	3_0 * 5_0 * 7_2
63	3*21	3_2 * 5_0 * 7_1

357 357O(n2)
357 3*3,3*5,3*7,5*5,5*7 35 53() 31325 37

1. res=1q3,q5,q7
2. q3,q5,q71*3,1*5,1*7
3. xres=x
4. x
   q3q3xq3q5q73*x,5*x,7*x q5q5xq5q75*x,7*x q7q7xq77*x
5. 35k

xq5q33*xq5

```
1  #include <iostream>
2  #include <queue>
3  using namespace std;
4
5  int mini(int a, int b) {
6      return a < b ? a : b;     // like
7  }
8
9  int mini(int a, int b, int c)  {
10     //return mini(a, b) < c : mini(a, b) : c;
11     return mini( mini(a, b), c );
12 }
13
14 int get_num(int val) {
15     if (val <= 0) return 0;
16     int res = 1, cnt = 1;
17     queue<int> q3, q5, q7;
18     q3.push(3); q5.push(5); q7.push(7);
19
20     for (; cnt < val; ++cnt) {
```

```
21          int v3 = q3.front();
22          int v5 = q5.front();
23          int v7 = q7.front();
24          res = mini(v3, v5, v7);
25          if (res == v7)
26              q7.pop();
27          else {
28              if (res == v5)
29                  q5.pop();
30              else {
31                  if (res == v3) {
32                      q3.pop();
33                      q3.push(3 * res);
34                  }
35              }
36              q5.push(5 * res);
37          }
38          q7.push(7 * res);
39      }
40      return res;
41  }
42
43  int main() {
44      for (int i = 1; i < 20; ++i)
45          cout << get_num(i) << endl;
46      return 0;
47  }
```

Chapter 11

Testing

11.1 Find the mistake(s) in the following code:

```
unsigned int i;
for (i = 100; i <= 0; --i)
printf(%d\n, i);
```

Solution:

i <= 0i >= 0OK iunsigned int 0 XD.

```
unsigned int i; //unsigned int0
for (i = 100; i <= 0; --i) //100<=0
                           //i>=0,i>=0
                           //i>01001
    printf(%d\n, i);
```

```
int i;
for (i = 100; i >= 0; --i)
    printf(%d\n, i);
1000
```

11.2 You are given the source to an application which crashes when it is run. After running it ten times in a debugger, you find it never crashes in the same place. The application is single threaded, and uses only the C standard library. What programming errors could be causing this crash? How would you test each one?

10 10C

1.
2.

() web

11.3 We have the following method used in a chess game: boolean canMoveTo(int x, int y) x and y are the coordinates of the chess board and it returns whether or not the piece can move to that position. Explain how you would test this method.

boolean canMoveTo(int x, int y) xy

a.

1. x y false
2. truefalse

b.

6()

- 6a
- ab
- d
- a()
- bd
-

11.4 How would you load test a webpage without using any test tools?

Solution:

1.
2.
3.
4.

IO

11.5 How would you test a pen?

Solution:

-
-
-
-
-
-
-
-
-
- 510

1.
2.
3. ()

11.6 How would you test an ATM in a distributed banking system?

ATM

Solution:

1. ATM()
2. ATM
3.

1.
2.
3.
4. ATM
5.
6. ATM

Chapter 12

System Design and Memory Limits

12.1 If you were integrating a feed of end of day stock price information (open, high, low,and closing price) for 5,000 companies, how would you do it? You are responsible for the development, rollout and ongoing monitoring and maintenance of the feed. Describe the different methods you considered and why you would recommend your approach. The feed is delivered once per trading day in a comma-separated format via an FTP site. The feed will be used by 1000 daily users in a web application.

5000 FTP 1000web

Solution:

FTP

1.
2.

return all stocks having open > N AND closing price < M (NM)

1.
2.

XML XML

```
<root>
<date value=2008-10-12>
<company name=foo>
<open>126.23</open>
<high>130.27</high>
<low>122.83</low>
<closingPrice>127.30</closingPrice>
</company>
<company name=bar>
<open>52.73</open>
<high>60.27</high>
<low>50.29</low>
<closingPrice>54.91</closingPrice>
</company>
</date>
<date value=2008-10-11> . . . </date>
</root>
```

1. XML
2.
3. XML

4. ()

12.2 How would you design the data structures for a very large social network (Facebook,LinkedIn, etc)? Describe how you would design an algorithm to show the connection, or path, between two people (e.g., Me -> Bob -> Susan -> Jason -> You).

FacebookLinkedIn -> Bob -> Susan -> Jason ->

Solution:

```
class Person {
    Person[] friends;
    // Other info
}
```

(BFS)BFS

OrkutFacebook Person PersonID

- idint machine_index = lookupMachineForUserID(id);
- machine_index
- Person friend = lookupFriend(machine_index);

5

1.
2.
3.
4.

javaC++

```
public class Server {
    ArrayList<Machine> machines = new ArrayList<Machine>();
}
public class Machine {
    public ArrayList<Person> persons = new ArrayList<Person>();
    public int machineID;
}

public class Person {
    private ArrayList<Integer> friends;
    private int ID;
    private int machineID;
    private String info;
    private Server server = new Server();

    public String getInfo() { return info; }

    public void setInfo(String info) {
        this.info = info;
```

```
20      }
21
22      public int[] getFriends() {
23          int[] temp = new int[friends.size()];
24          for (int i = 0; i < temp.length; i++) {
25              temp[i] = friends.get(i);
26          }
27          return temp;
28      }
29
30      public int getID() { return ID; }
31      public int getMachineID() { return machineID; }
32      public void addFriend(int id) { friends.add(id); }
33
34      // Look up a person given their ID and Machine ID
35
36      public Person lookUpFriend(int machineID, int ID) {
37          for (Machine m : server.machines) {
38              if (m.machineID == machineID) {
39                  for (Person p : m.persons) {
40                      if (p.ID == ID){
41                          return p;
42                      }
43                  }
44              }
45          }
46          return null;
47      }
48
49      // Look up a machine given the machine ID
50
51      public Machine lookUpMachine(int machineID) {
52          for (Machine m:server.machines) {
53              if (m.machineID == machineID)
54                  return m;
55          }
56          return null;
57      }
58
59      public Person(int iD, int machineID) {
60          ID = iD;
61          this.machineID = machineID;
62      }
63  }
```

12.3 Given an input file with four billion integers, provide an algorithm to generate an integer which is not contained in the file. Assume you have 1 GB of memory.

FOLLOW UP

What if you have only 10 MB of memory?

40 1GB

10MB

Solution:

40

40 * 10$\hat{8}$ * 4B = 16GB (2) 15151 Bit Map

http://blog.csdn.net/hguisu/article/details/7880288

Bit Map

40 * 10$\hat{8}$ b = 5 * 10$\hat{8}$ B = 0.5GB

1GBBit Map (-11)0x7FFFFFFF 0x7FFFFFFF

1 0

```cpp
#include <iostream>
#include <cstdio>
using namespace std;

int main() {
    freopen("ch1212.3.in", "r", stdin);

    int int_len = sizeof(int) * 8;
    int bit_len = 0xFFFFFFFF / int_len;
    int * bit = new int[bit_len];
    int v;

    while( scanf("%d", &v) != EOF ) {
        bit[v/int_len] |= 1 << (v % int_len);
    }
    bool found = false;
    for (int i = 0; i < bit_len; ++i) {
        for (int j = 0; j < int_len; ++j) {
            if ( (bit[i] & (1<<j)) == 0 ) {
                cout << i*int_len + j << endl;
                found = true;
                break;
            }
        }
        if (found) break;
    }
    delete[] bit;
    fclose(stdin);
    return 0;
}
```

10MBBit Map 1000099921000 1999 1 (1000)
Bit Map 1(0blocksize) 0

```cpp
#include <iostream>
#include <cstdio>
using namespace std;

int main(){
    freopen("12.3.in", "r", stdin);// 20000 number
    int int_len = sizeof(int) * 8;
    int totalnum = 20000;
    int blocksize = 2000;
    int blocknum = totalnum / blocksize;
    int* block = new int[blocknum];
    int* bit = new int[blocksize/int_len+1];
    int v;
    while(scanf("%d", &v) != EOF){
        ++block[v/blocksize];
    }
    fclose(stdin);
    int start;
    for(int i=0; i<blocknum; ++i){
        if(block[i] < blocksize){
            start = i * blocksize;
            break;
        }
    }
    freopen("12.3.in", "r", stdin);
```

```
26      while(scanf("%d", &v) != EOF){
27          if(v>=start && v<start+blocksize){
28              v -= start; // make v in [0, blocksize)
29              bit[v/int_len] |= 1<<(v%int_len);
30          }
31      }
32
33      bool found = false;
34      for(int i=0; i<blocksize/int_len+1; ++i){
35          for(int j=0; j<int_len; ++j){
36              if((bit[i] & (1<<j)) == 0){
37                  cout<<i*int_len+j+start<<endl;
38                  found = true;
39                  break;
40              }
41          }
42          if(found) break;
43      }
44
45      delete[] block;
46      delete[] bit;
47      fclose(stdin);
48      return 0;
49  }
```

12.4 You have an array with all the numbers from 1 to N, where N is at most 32,000. The array may have duplicate entries and you do not know what N is. With only 4KB of memory available, how would you print all duplicate elements in the array?

1NN32000.(2)N4KB

4KB4 * 210 * 832000Bit Map Bit Map 32int32 32sizeof(int)*8

Solution:

```
1   #include <iostream>
2   using namespace std;
3
4   class Bitmap{
5   private:
6       int *bits;
7   public:
8       Bitmap(int size = 32) {
9           bits = new int[size/32+1];
10      }
11      ~Bitmap() {
12          delete[] bits;
13      }
14      bool get(int pos) { // true if bit is 1, else: false
15       // return (bits[pos/32] & (1<<pos)) != 0;  // result equavilent
16          return (bits[pos/32] & (1<<(pos & 0x1f))) != 0;
17
18      }
19      void set(int pos) {
20          bits[pos/32] |= ( 1 << (pos & 0x1f) );
21      }
22  };
23
24  void print_duplicates(int a[], int n, int bitsize) {
25      Bitmap bm(bitsize);
26      for (int i = 0; i < n; ++i) {
27          if ( bm.get(a[i]-1) ) // bitmap starts at 0, number starts at 1
28              cout << a[i] << endl;
```

```
29            else
30                bm.set( a[i]-1 );
31        }
32    }
33
34    int main() {
35        int a[] = {1, 2, 3, 4, 5, 32000, 7, 8, 9, 10, 11, 1, 2, 13, 15, 16, 32000, 11, 5, 8};
36        print_duplicates(a, 20, 32000);
37        return 0;
38    }
```

12.5 If you were designing a web crawler, how would you avoid getting into infinite loops?

Solution:
python
AB BA
DFS()BFS()

12.6 You have a billion urls, where each is a huge page. How do you detect the duplicate documents?
10urlurl
Solution:

1. hash urlurl
2. 10 O(n2)

1.
2. url url url

1.
   - 4
   - url3030
   - url34
2. 34 * 10 = 31.6GB

1. /
2.
3. n
   - v
   - v%n
   - v/n

12.7 You have to design a database that can store terabytes of data. It should support efficient range queries. How would you do it?
TB()
Solution:
ab
B+ B+ B+ O(h)h () ()
(B+) ()

B+ OK 25k60k
252559 251544 2537 (44)(51,59)(60)
B+XD

Chapter 13

C++

13.1 Write a method to print the last K lines of an input file using C++.
C++k
Solution:
NN-K
k k k k+11 ii%k k0i%k

```
#include <iostream>
#include <fstream>
using namespace std;

void printLastKLines(ifstream &fin, int k) {
    string line[k];
    int lines = 0;
    string tmp;

 // while (fin.get_line() != EOF) {   // my original, mistake
 // while ( !fin.eof() ) {    // WRONG
    while ( getline(fin, tmp) ) {
        line[lines % k] = tmp;
        ++lines;
    }

    /* // my original
    if (lines % k == 0)
        for (int i = 0; i < k; ++k)
            cout << line[i] << endl;
    else {
        for (int i = lines%k; i < k; ++i)
            cout << line[i] << endl;
        for (int i = 0; i < lines % k; ++i)
            cout << line[i] << endl;
    } */

    int start, cnt;
    if (lines < k) {
        start = 0;
        cnt = k;
    } else {
        start = lines % k;
        cnt = k;
    }
```

```
36      for (int i = 0; i < cnt; ++i)
37          cout << line[ (start+i) % k ] << endl;
38  }
39
40  int main() {
41      ifstream fin("ch1313.1.in");
42      int k = 4;
43      printLastKLines(fin, k);
44      fin.close();
45      return 0;
46  }
```

while

```
1   while(!fin.eof()){
2       getline(fin, line[lines%k]);
3       ++lines;
4   }
```

eof()fintrue getline eof()false getlineeof()true linek-1 printLastKLines
EOF

13.2 Compare and contrast a hash table vs. an STL map. How is a hash table implemented?If the number of inputs is small, what data structure options can be used instead of a hash table?
STL map
Solution:

a. STL map
O(1) ()
STLmap/ O(logn) STLmap

1.
2.
3.
4.

b.

1.
2. (chaining) (probing)
3. ()/()

c.
STL mapO(logn)

13.3 How do virtual functions work in C++?
C++
Solution:
virtual C++
vptr

```
1   class Shape{
2   public:
3       int edge_length;
4       virtual int circumference () {
5           cout<<"Circumference␣of␣Base␣Class\n";
6           return 0;
7       }
8   };
9
10  class Triangle: public Shape{
11  public:
12      int circumference () {
```

```
13          cout<<"Circumference of Triangle Class\n";
14          return 3 * edge_length;
15      }
16  };
17
18  int main(){
19      Shape *x = new Shape();
20      x->circumference(); // prints Circumference of Base Class
21      Shape *y = new Triangle();
22      y->circumference(); // prints Circumference of Triangle Class
23      return 0;
24  }
```

circumferenceShape C++

13.4 What is the difference between deep copy and shallow copy? Explain how you would use each.

Solution:

```
1  struct Test{
2      char *ptr;
3  };
4
5  void shallow_copy(Test &src, Test &dest){
6      dest.ptr = src.ptr;
7  }
8
9  void deep_copy(Test &src, Test &dest){
10     dest.ptr = (char*)malloc(strlen(src.ptr) + 1);
11     memcpy(dest.ptr, src.ptr);
12 }
```

0

13.5 What is the significance of the keyword volatile in C?

C"volatile"()

Solution:

volatile"" volatile ()

volatile

```
1  volatile int x;
2  int volatile x;
```

volatile():

```
1  volatile int *x;
2  int volatile *x
```

volatilevolatile:

```
1  int* volatile x;
```

volatilevolatile():

```
1  volatile int * volatile x;
2  int volatile * volatile x;
```

volatileconstvolatile* *

volatile

```
int opt = 1;
void Fn(void){
    start:
        if (opt == 1) goto start;
        else break;
}
```

```
void Fn(void){
    start:
        int opt = 1;
        if (true)
            goto start;
}
```

optif optvolatile opt volatile

13.6 What is name hiding in C++?

C++

Solution:

C++C++ ()

```
class FirstClass{
public:
    virtual void MethodA(int);
    virtual void MethodA(int, int);
};

void FirstClass::MethodA(int i){
    cout<<"ONE"<<endl;
}

void FirstClass::MethodA(int i, int j){
    cout<<"TWO"<<endl;
}
```

()

```
class SecondClass : public FirstClass{
public:
    void MethodA(int);
};

void SecondClass::MethodA(int i){
    cout<<"THREE"<<endl;
}

int main (){
    SecondClass a;
    a.MethodA(1);
    a.MethodA(1, 1);
    return 0;
}
```

main2MethodA MethodA
2MethodA MethodA

13.7 Why does a destructor in base class need to be declared virtual?

Solution:

1.
2.

(2)

```
class Base{
public:
    Base() { cout<<"Base Constructor"<<endl; }
    ~Base() { cout<<"Base Destructor"<<endl; }
};

class Derived: public Base{
public:
    Derived() { cout<<"Derived Constructor"<<endl; }
    ~Derived() { cout<<"Derived Destructor"<<endl; }
};

int main(){
    Base *p = new Derived();
    delete p;
    return 0;
}
```

- Base Constructor
- Derived Constructor
- Base Destructor

- Base Constructor
- Derived Constructor
- Derived Destructor
- Base Destructor

Effective C++

13.8 Write a method that takes a pointer to a Node structure as a parameter and returns a complete copy of the passed-in data structure. The Node structure contains two pointers to other Node structures.

Node NodeNode

Solution:

Node

```
typedef map<Node*, Node*> NodeMap;

Node* copy_recursive(Node *cur, NodeMap &nodeMap){
    if(cur == NULL){
        return NULL;
    }
    NodeMap::iterator i = nodeMap.find(cur);
    if (i != nodeMap.end()){
        // weve been here before, return the copy
        return i->second;
    }
    Node *node = new Node;
    nodeMap[cur] = node; // map current node before traversing links
    node->ptr1 = copy_recursive(cur->ptr1, nodeMap);
    node->ptr2 = copy_recursive(cur->ptr2, nodeMap);
    return node;
```

```
}

Node* copy_structure(Node* root){
    NodeMap nodeMap; // we will need an empty map
    return copy_recursive(root, nodeMap);
}
```

13.9 Write a smart pointer (smart_ptr) class.
(smart_ptr)
Solution:
() (dangling pointers)(memory leaks) (0)

```
#include <iostream>
#include <cstdlib>
using namespace std;

template <typename T>

class SmartPointer {
private:
    void clear() {
        delete ref;
        free(ref_count);
        ref = NULL;  // prevent it from potentially becoming ''dangling pointer''
        ref_count = NULL;
    }
protected:
    T *ref;
    unsigned *ref_count;
public:
    SmartPointer(T* ptr) {
        ref = ptr;
        ref_count = (unsigned*)malloc(sizeof(unsigned));
        *ref_count = 1;
    }
    SmartPointer(SmartPointer<T> &sptr) {
        ref = sptr.ref;
        ref_count = sptr.ref_count;
        ++*ref_count;
    }
    SmartPointer<T>& operator = (SmartPointer<T> &sptr) {
        if (this != &sptr) {
            if (--*ref_count == 0) {
                clear();
                cout << "operator= clear" << endl;
            }
            ref = sptr.ref;
            ref_count = sptr.ref_count;
            ++*ref_count;
        }
        return *this;
    }
    ~SmartPointer() {
        if (--*ref_count == 0) {
            clear();
            cout << "destructor clear" << endl;
        }
    }

    T getValue() { return *ref; }
```

```
49 };
50
51 int main() {
52     int *ip1 = new int();
53     *ip1 = 11111;
54     int *ip2 = new int();
55     *ip2 = 22222;
56     SmartPointer<int> sp1(ip1), sp2(ip2);
57     SmartPointer<int> spa = sp1;
58     sp2 = spa; // get different result when commented out
59     return 0;
60 }
```

0 10

mainsp2 = spa

- destructor clear
- destructor clear

main

- operator= clear
- destructor clear

sp2 main

CTCIsp2 = spa

- destructor clear

(ip1)

Chapter 14

Java

14.1 In terms of inheritance, what is the effect of keeping a constructor private?

14.2 In Java, does the finally block gets executed if we insert a return statement inside the try block of a try-catch-finally?

14.3 What is the difference between final, finally, and finalize?

14.4 Explain the difference between templates in C++ and generics in Java.

14.5 Explain what object reflection is in Java and why it is useful.

14.6 Suppose you are using a map in your program, how would you count the number of times the program calls the put() and get() functions?

Chapter 15

Databases

15.1 Write a method to find the number of employees in each department.

SQL

Solution:

0

```
1 select Dept_Name, Departments.Dept_ID, count(*) as 'num_employees'
2 from Departments
3 left join Employees
4 on Employees.Dept_ID = Departments.Dept_ID
5 group by Departments.Dept_ID, Dept_Name
```

15.2 What are the different types of joins? Please explain how they differ and why certain types are better in certain situations.

SQL

Solution:

(JOIN) ()

Regular Beverages: Name, Code

Calorie-Free Beverages: Code, Name

RBCFB

1. (INNER JOIN) 31COCACOLA2PEPSISQL

```
1 SELECT * from RB INNER JOIN CFB ON RB.Code = CFB.Code
```

2. (OUTER JOIN)

2.1 (LEFT OUTER JOIN)(LEFT JOIN) NULL 43BUDWEISERSQL

```
1 SELECT * from RB LEFT OUTER JOIN CFB ON RB.Code = CFB.Code
```

2.2 (RIGHT OUTER JOIN)(RIGHT JOIN) 53FRESCAWATERSQL

```
1 SELECT * from RB RIGHT OUTER JOIN CFB ON RB.Code = CFB.Code
```

2.3 (FULL OUTER JOIN) NULL 6SQL

```
1 SELECT * from RB FULL OUTER JOIN CFB ON RB.Code = CFB.Code
```

OK (SQL)

http://zh.wikipedia.org/zh/%E8%BF%9E%E6%8E%A5_(SQL)

15.3 What is denormalization? Explain the pros and cons.

Solution:

(JOIN)JOIN (DBMS) DBMS SQL(SQL Server)(Oracle)
DBMS (SELECT) (INSERT, UPDATEDELETE)

http://en.wikipedia.org/wiki/Denormalization

15.4 Draw an entity-relationship diagram for a database with companies, people, and professionals (people who work for companies).

(ER)(companies)(people) (professionals)

Solution:

(people)(professionals) professionalspeopleISA(is a) professionalspeople

peopleprofessional

professional() professional (work for)" professional professionalcompanies

ER http://hawstein.com/posts/15.4.html

15.5 Imagine a simple database storing information for students grades. Design what this database might look like, and provide a SQL query to return a list of the honor roll students (top 10%), sorted by their grade point average.

SQL(10%)GPA

Solution:

(Students)(Courses) (CourseEnrollment)ID ID ()

SQL

```
1  SELECT StudentName, GPA
2  FROM (
3  SELECT  top 10 percent Avg(CourseEnrollment.Grade) AS GPA,
4   CourseEnrollment.StudentID
5  FROM CourseEnrollment
6  GROUP BY CourseEnrollment.StudentID
7  ORDER BY Avg(CourseEnrollment.Grade)) Honors
8  INNER JOIN Students ON Honors.StudentID = Students.StudentID
```

Chapter 16

Low Level

16.1 Explain the following terms: virtual memory, page fault, thrashing.

Solution:

RAM
http://zh.wikipedia.org/wiki/
(Page)() page page fault
http://zh.wikipedia.org/wiki/
/
http://en.wikipedia.org/wiki/Thrash_(computer_science)

16.2 What is a Branch Target buffer? Explain how it can be used in reducing bubble cycles in cases of branch misprediction.

16.3 Describe direct memory access (DMA). Can a user level buffer / pointer be used by kernel or drivers?

16.4 Write a step by step execution of things that happen after a user presses a key on the keyboard. Use as much detail as possible.

16.5 Write a program to find whether a machine is big endian or little endian.

Solution:
http://zh.wikipedia.org/wiki/%E5%AD%97%E8%8A%82%E5%BA%8F

```
#define BIG_ENDIAN 0
#define LITTLE_ENDIAN 1
int TestByteOrder(){
    short int word = 0x0001;
    char *byte = (char *) &word;
    return (byte[0] ? LITTLE_ENDIAN : BIG_ENDIAN);
}
```

16.6 Discuss how would you make sure that a process doesnt access an unauthorized part of the stack.

16.7 What are the best practices to prevent reverse engineering of DLLs?

16.8 A device boots with an empty FIFO queue. In the first 400 ns period after startup, and in each subsequent 400 ns period, a maximum of 80 words will be written to the queue. Each write takes 4 ns. A worker thread requires 3 ns to read a word, and 2 ns to process it before reading the next word. What is the shortest depth of the FIFO such that no data is lost?

16.9 Write an aligned malloc & free function that takes number of bytes and aligned byte (which is always power of 2) EXAMPLE
align_malloc (1000,128) will return a memory address that is a multiple of 128 and that points to memory of size 1000 bytes.
aligned_free() will free memory allocated by align_malloc.

16.10 Write a function called my2DAlloc which allocates a two dimensional array. Minimize the number of calls to malloc and make sure that the memory is accessible by the notation arr[i][j].

my2DAllocmalloc arr[i][j]ij

Solution:

```
int** My2DAlloc(int rows, int cols){
    int **arr = (int**)malloc(rows*sizeof(int*));
    for(int i=0; i<rows; ++i)
        arr[i] = (int*)malloc(cols*sizeof(int));
    return arr;
}
```

(rows+1)mallocmalloc malloc
rows(int*) rows*cols(int) OK

```
int header = rows * sizeof(int*);
```

rows*cols

```
int data = rows * cols * sizeof(int);
```

```
int **arr = (int**)malloc(header + data);
```

rows * sizeof(int*)arrint** rowsint*

```
int *buf = (int*)(arr + rows);
```

bufcolsarr

```
for(int i=0; i<rows; ++i)
    arr[i] = buf + i * cols;
```

```
int** My2DAlloc1(int rows, int cols){
    int header = rows * sizeof(int*);
    int data = rows * cols * sizeof(int);
    int **arr = (int**)malloc(header + data);
    int *buf = (int*)(arr + rows);
    for(int i=0; i<rows; ++i)
        arr[i] = buf + i * cols;
    return arr;
}
```

malloc arr[i][j]

Chapter 17

Networking

17.1 Explain what happens, step by step, after you type a URL into a browser. Use as much detail as possible.
URL
Solution:

1. DNSURLIP
2. DNSIP
3. IPweb80TCP
4. html
5. HTML
6.

12()()IP DNSIP(DNS) IPDNS
IP IP IP

What really happens when you navigate to a URL
http://igoro.com/archive/what-really-happens-when-you-navigate-to-a-url/
URL
http://www.cnblogs.com/panxueji/archive/2013/05/12/3073924.html

17.2 Explain any common routing protocol in detail. For example: BGP, OSPF, RIP.
BGPOSPFRIP
Solution:

BGP : Border Gateway Protocol ()
BGPBGP BGP
BGPIP BGP BGP BGP
http://www.livinginternet.com/i/iw_route_egp_bgp.htm

RIP : Routing Information Protocol ()
RIPIGP()
RIP
http://www.livinginternet.com/i/iw_route_igp_rip.htm

OSPF : Open Shortest Path First ()
OSPFIGPRIP
OSPFRIPRIP OSPFOSPF

1. "hello" "hello"
2.

3.

http://www.livinginternet.com/i/iw_route_igp_ospf.htm

17.3 Compare and contrast the IPv4 and IPv6 protocols.

IPv4IPv6

Solution:

IPv4IPv6IPv4 IPv6

- IPv4432IPv6 128
- IPv44,294,967,296IPIPv634, 000, 000, 000, 000, 000, 000, 000, 000, 000, 000, 000, 000, 000IP (34360)
- IPv4IPABCDE ABC3IPDE IP()
- IPv63
  1. IPv6
  2. IPv6
  3. () ()IPv6
- IPv4239.255.255.255, 255.255.255.0
- IPv6:816 2001:cdba:0000:0000:0000:0000:3257:9652

(IP)IPv6

1.
2.
3.
4. ISP
5.
6.
7.
8.

17.4 What is a network / subnet mask? Explain how host A sends a message / packet to host B when: (a) both are on same network and (b) both are on different networks. Explain which layer makes the routing decision and how.

/A/B(a) AB (b) AB

Solution:

/IP

IP

IP152.210.011.002B

- 11111111.11111111.00000000.00000000
- IP 10011000.11010101.00001011.00000010

IP

- 10011000.11010101.00000000.00000000 (152.213.0.0)
- 00001011.00000010

18(525 > 18) 5

- 11111111.11111111.11111000.00000000 (255.255.248.0)
- IP 10011000.11010101.00001011.00000010

IP

- 10011000.11010101.00001000.00000000 (152.213.8.0)

A/B

17.5 What are the differences between TCP and UDP? Explain how TCP handles reliable delivery (explain ACK mechanism), flow control (explain TCP sender / receiver window) and congestion control.

TCPUDPTCP(ACK) (TCP/)

Solution:

TCP ()TCP TCP

- TCP
- TCP
- TCP

UDP ()UDPUDP/ UDP

-
-
-

C. TCP(ACK)(TCP/)

TCP ACKTCPTCP

TCPTCP64KB 64KB 64KB 32KB(32KB)32KB64KB (32KB32KB)

D. : TCP TahoeRenoTCP TCP TCPTCP (MSS)2 1MSS(RTT)

TCP http://www.cnblogs.com/fll/archive/2008/06/10/1217013.html

Chapter 18

Threads and Locks

18.1 Whats the difference between a thread and a process?

Solution:

()

1. CPU
2.
3.
4.
5.

18.2 How can you measure the time spent in a context switch?

Solution:

()CPU () ()

P1P2
P1P2P1P2 P1NTime_Stamp(P1_N)P2 1Time_Stamp(P2_1) Time_Stamp(P2_1) - Time_Stamp(P1_N)

T T - ()

18.3 Implement a singleton design pattern as a template such that, for any given class Foo, you can call Singleton::instance() and get a pointer to an instance of a singleton of type Foo. Assume the existence of a class Lock which has acquire() and release() methods. How could you make your implementation thread safe and exception safe?

FooSingleton::instance() FooLockacquire() release()

Solution:

```
1 #include <iostream>
2 using namespace std;
3
4 /*  */
```

```
5  class Lock {
6  public:
7      Lock() {  /*  */ }
8      ~Lock() { /*  */ }
9      void AcquireLock() { /*  */ }
10     void ReleaseLock() { /*  */ }
11 };
12
13 //
14 template <typename T>
15 class Singleton{
16 private:
17     static Lock lock;
18     static T* object;
19 protected:
20     Singleton() { };
21 public:
22     static T* Instance();
23 };
24
25 template <typename T>
26 Lock Singleton<T>::lock;
27
28 template <typename T>
29 T* Singleton<T>::object = NULL;
30
31 template <typename T>
32 T* Singleton<T>::Instance(){
33     if (object == NULL){// object
34         lock.AcquireLock();
35         //if
36         //objectobjectNULL
37         //
38         if (object == NULL){
39             object = new T;
40         }
41         lock.ReleaseLock();
42     }
43     return object;
44 }
45 class Foo{
46
47 };
48 int main(){
49     Foo* singleton_foo = Singleton<Foo>::Instance();
50     return 0;
51 }
```

offer2Singleton

18.4 Design a class which provides a lock only if there are no possible deadlocks.

18.5 Suppose we have the following code:

```
1  class Foo{
2  public:
3      A(.....); /*If A is called, a new thread will be created and
4                  the corresponding function will be executed. */
5      B(.....); /*same as above */
6      C(.....); /*same as above */
```

```
};
Foo f;
f.A(.....);
f.B(.....);
f.C(.....);
```

i) Can you design a mechanism to make sure that B is executed after A, and C is executed after B?

ii) Suppose we have the following code to use class Foo. We do not know how the threads will be scheduled in the OS.

```
Foo f;
f.A(.....); f.B(.....); f.C(.....);
f.A(.....); f.B(.....); f.C(.....);
```

Can you design a mechanism to make sure that all the methods will be executed in sequence?

```
class Foo{
public:
    A(.....); /*A*/
    B(.....); /**/
    C(.....); /**/
};
Foo f;
f.A(.....);
f.B(.....);
f.C(.....);
```

i) BACB

ii)

```
Foo f;
f.A(.....); f.B(.....); f.C(.....);
f.A(.....); f.B(.....); f.C(.....);
```

Solution:

0As_a1B Bs_b1CABC AAA3 s_a=3Bs_a=2,s_b=1Cs_a=2,s_b=0 Bs_a=1,s_b=1AAABCB

```
Semaphore s_a(0);
Semaphore s_b(0);
A {
    s_a.release(1); // s_a1
}
B {
    s_a.acquire(1); // s_a1
    s_b.release(1); // s_b1
}
C {
    s_b.acquire(1); // s_b1
}
```

ABCABCABC0 BACB ACABC

```
Semaphore s_a(0);
Semaphore s_b(0);
Semaphore s_c(1);
A {
    s_c.acquire(1);
    /***/
    s_a.release(1);
```

```
8  }
9  B {
10     s_a.acquire(1);
11     /****/
12     s_b.release(1);
13 }
14 C {
15     s_b.acquire(1);
16     /******/
17     s_c.release(1);
18 }
```

18.6 You are given a class with synchronized method A, and a normal method C. If you have two threads in one instance of a program, can they call A at the same time? Can they call A and C at the same time?

Chapter 19

Moderate

19.1 Write a function to swap a number in place without temporary variables.

Solution:

swap

```
// 1
void swap(int &a, int &b){
    b = a - b;
    a = a - b;
    b = a + b;
}
// 2
void swap(int &a, int &b){
    a = a ^ b;
    b = a ^ b;
    a = a ^ b;
}
```

swap2 swap swapswap

```
swap(a[i], a[j]); // i==j
```

swapij 0

swap

```
void swap(int &a, int &b){
    int t = a;
    a = b;
    b = t;
}
```

19.2 Design an algorithm to figure out if someone has won in a game of tic-tac-toe.

Solution:

HasWon

HasWon

339 = 19683 O(1) 30()1()2() 08310

- 1 2 2
- 2 1 1
- 2 0 1
- :
- 122211201=1*3$\hat{8}$ + 2*3$\hat{7}$ +...+ 0*3$\hat{1}$ + 1*3$\hat{0}$

1 1 char

HasWon

HasWon

n*n

```
#include <iostream>
using namespace std;

enum Check {ROW, COLUMN, DIAGONAL, REDIAGONAL};

int CheckRowColumn(int board[], int n, Check check) {
    int type = 0;
    for (int i = 0; i < n; ++i) {
        bool found = true;
        for (int j = 0; j < n; ++j) {
            int k = 0;
            if (check == ROW)
                k = i * n + j;
            else
                k = i + j * n;

            if (j == 0)
                type = board[k];
            else if ( board[k] != type ) {
                found = false;
                break; //
            }
        }
        if (found) return type;
    }
    return 0;
}

int CheckDiagonal(int board[], int n, Check check) {
    int type = 0;
    bool found = true;
    //
    for (int i = 0; i < n; ++i) {
        int k = 0;
        if (check == DIAGONAL)
            k = i + i * n;
        else
            k = i + (n-1-i) * n;
        if (i == 0)
            type = board[k];
        else if (board[k] != type) {
            found = false;
            break;
        }
    }
    if (found) return type;
    return 0;
}

int HasWon(int board[], int n) {
    int type = 0;
    type = CheckRowColumn(board, n, ROW);
    if (type != 0) return type;
    type = CheckRowColumn(board, n, COLUMN);
    if (type != 0) return type;
```

```
56      type = CheckDiagonal(board, n, DIAGONAL);
57      if (type != 0) return type;
58      type = CheckDiagonal(board, n, REDIAGONAL);
59      if (type != 0) return type;
60      return 0;
61  }
62
63  int main() {
64      int n = 3; // 3 * 3
65      int board[] = {2, 2, 1, 2, 1, 1, 1, 2, 0};
66      int type = HasWon(board, n);
67      if (type != 0)
68          cout << type << " won!" << endl;
69      else
70          cout << "Nobody won!" << endl;
71      return 0;
72  }
```

19.3 Write an algorithm which computes the number of trailing zeros in n factorial.

n0

Solution:

n0 n n0

n0525*2=10n 52255

- 5!, 1*5, 15
- 10!, 1*5,2*5, 25
- 15!, 1*5,2*5,3*5, 35
- 20!, 1*5,2*5,3*5,4*5, 45
- 25!, 1*5,2*5,3*5,4*5,5*5, 65
- ...

nn51n15n5n 55n5 5 250 25555,10,15,20,2555 n=25/5=5n51255 55

```
1   int NumZeros(int n){
2       if(n < 0) return -1;
3       int num = 0;
4       while((n /= 5) > 0){
5           num += n;
6       }
7       return num;
8   }
```

19.4 Write a method which finds the maximum of two numbers. You should not use if-else or any other comparison operator.

Input: 5, 10

Output: 10

if-else

Solution:

if-else

- If a > b, return a; else, return b.
- If (a - b) < 0, return b; else, return a.
- If (a - b) < 0, k = 1; else, k = 0. return a - k * (a - b).
- z = a - b. kzreturn a - k * z.

aba-b0a-k*z = aab a-b1a-k*z = b

kz(01)ca,bc[k]

```
#include <iostream>
using namespace std;

int Max1(int a, int b) {
    int c[2] = {a, b};
    int z = a - b;
    z = (z >> 31) & 1;
    return c[z];
}

int Max2(int a, int b) {
    int z = a - b;
    int k = (z >> 31) & 1;
    return a - k*z;
}

int main() {
    int a = 5, b = 10;
    cout << Max1(a, b) << endl;
    cout << Max2(a, b) << endl;
    return 0;
}
```

19.5 The Game of Master Mind is played as follows:

The computer has four slots containing balls that are red (R), yellow (Y), green (G) or blue (B). For example, the computer might have RGGB (e.g., Slot #1 is red, Slots #2 and #3 are green, Slot #4 is blue).

You, the user, are trying to guess the solution. You might, for example, guess YRGB.When you guess the correct color for the correct slot, you get a hit. If you guess a color that exists but is in the wrong slot, you get a pseudo-hit. For example, the guess YRGB has 2 hits and one pseudo hit.

For each guess, you are told the number of hits and pseudo-hits. Write a method that, given a guess and a solution, returns the number of hits and pseudo hits.

Master Mind

444(R)(Y)(G)(B) RGGB

YRGB hit34(GB)2hit pseudo-hit Rpseudo-hit

hitspseudo-hits hitspseudo-hits

Solution:

pseudo-hits pseudo-hit RYGBYRRRhits0pseudo-hits YYpseudo-hits3R 1pseudo-hits3 CTCI33R 31RR

```
#include <iostream>
#include <cstring>
using namespace std;

struct Result {
    int hits;
    int pseudo_hits;
};

Result Estimate(const char* solution, const char* guess) {
    Result res;
    res.hits = 0;
    res.pseudo_hits = 0;
    int solution_mask = 0;

    for (int i = 0; i < 4; ++i)
        solution_mask |= 1 << (solution[i] - 'A');

    for (int i = 0; i < 4; ++i) {
```

```
20          if (guess[i] == solution[i])
21              ++res.hits;
22          else if ( solution_mask & (1 << (guess[i]-'A')) )
23              ++res.pseudo_hits;
24      }
25      return res;
26  }
27
28  int Min(int a, int b) {
29      return a < b ? a : b;
30  }
31
32  Result Estimate1(const char* solution, const char* guess) {
33      Result res;
34      res.hits = 0;
35      res.pseudo_hits = 0;
36      int num = 26 + 5;
37      int guess_count[num], solution_count[num];
38      memset(guess_count, 0, sizeof(guess_count));
39      memset(solution_count, 0, sizeof(solution_count));
40
41      for (int i = 0; i < 4; ++i) {
42          if (guess[i] == solution[i])
43              ++res.hits;
44          ++guess_count[(int)(guess[i]-'A')];
45          ++solution_count[(int)(solution[i]-'A')];
46      }
47      char color[] = "RGBY";
48      for (int i = 0; i < 4; ++i) {
49          int idx = (int)(color[i] - 'A');
50          res.pseudo_hits += Min(guess_count[idx], solution_count[idx]);
51      }
52      res.pseudo_hits -= res.hits;
53      return res;
54  }
55
56  int main() {
57      char solution[] = "RYGB";
58      char guess[] = "YRRR";
59      Result res = Estimate(solution, guess);
60      cout << res.hits << "␣" << res.pseudo_hits << endl;
61      Result res1 = Estimate1(solution, guess);
62      cout << res1.hits << "␣" << res1.pseudo_hits << endl;
63
64      return 0;
65  }
```

19.6 Given an integer between 0 and 999,999, print an English phrase that describes the integer (eg, One Thousand, Two Hundred and Thirty Four).

19.7 You are given an array of integers (both positive and negative). Find the continuous sequence with the largest sum. Return the sum.

EXAMPLE

Input: 2, -8, 3, -2, 4, -10

Output: 5 (i.e., 3, -2, 4)

()

: 2, -8, 3, -2, 4, -10

: 5 (, 3, -2, 4)

Solution:

maxsum cursum3 cursum0 cursum cursum0cursum maxsummaxsum

```
#include <iostream>
using namespace std;

bool g_Invalid = false;

int GetMaxSum(int a[], int n) {
    if (a == NULL || n <= 0) {
        g_Invalid = true;
        return 0;
    }
    g_Invalid = false;

    int max_sum = 1 << 31; // Min Int
    int cur_sum = 0;
    for (int i = 0; i < n; ++i) {
        if (cur_sum <= 0)
            cur_sum = a[i];
        else if (cur_sum > 0)
            cur_sum += a[i];

        if (cur_sum > max_sum)
            max_sum = cur_sum;
    }
    return max_sum;
}

int main() {
    int a[] = {2, -8, 3, -2, 4, -10};
    int max_sum = GetMaxSum(a, 6);
    if (g_Invalid)
        cout << "Invalid Input!" << endl;
    else
        cout << max_sum << endl;
    return 0;
}
```

19.8 Design a method to find the frequency of occurrences of any given word in a book.

Solution:
19.2 O(1)

O(n) 2 4

O(1)

19.9 Since XML is very verbose, you are given a way of encoding it where each tag gets mapped to a pre-defined integer value. The language/grammar is as follows:

- Element –> Element Attr* END Element END [aka, encode the element tag, then its attributes, then tack on an END character, then encode its children, then another end tag]
- Attr –> Tag Value [assume all values are strings]
- END –> 01
- Tag –> some predefined mapping to int
- Value –> string value END

Write code to print the encoded version of an xml element (passed in as string).
FOLLOW UP
Is there anything else you could do to (in many cases) compress this even further?

19.10 Write a method to generate a random number between 1 and 7, given a method that generates a random number between 1 and 5 (i.e., implement rand7() using rand5()).

1517 (rand5()rand7())

Solution:

rand51,2,3,4,5rand71,2,3,4,5,6,7 rand567rand5rand7 rand7rand5

rand715

```
int Rand5(){
    int x = ~(1<<31); // max int
    while(x > 5)
        x = Rand7();
    return x;
}
```

1515 Rand7(1/7) Rand515 151/5OKRand51 while15 1Rand7n 111/72111567 (2/7)*(1/7)1

- P(x=1)=1/7 + (2/7) * 1/7 + (2/7)$\hat{}$2 * 1/7 + (2/7)$\hat{}$3 * 1/7 + ...
- =1/7 * (1 + 2/7 + (2/7)$\hat{}$2 + ...) //
- =1/7 * 1 / (1 - 2/7)
- =1/7 * 7/5
- =1/5

Rand51,2,3,4,5(1/5) a > bRandaRandb Randa1aRandb1b

```
// a > b
int Randb(){
    int x = ~(1<<31); // max int
    while(x > b)
        x = Randa();
    return x;
}
```

Rand5Rand7Rand5 Randaa > 7 Randa1a

- Rand5() + Rand5() - 1

191 Rand5()1(1, 1)2(1, 2)(2, 1)6

- 5 * (Rand5() - 1) + Rand5()

Rand51510450,5,10,15,20 Rand5()1,2,3,4,5125 125OK

```
int Rand7(){
    int x = ~(1<<31); // max int
    while(x > 7)
        x = 5 * (Rand5() - 1) + Rand5() // Rand25
    return x;
}
```

while Rand2512517while whilex25257 x > 21x121 1-7

```
int Rand7(){
    int x = ~(1<<31); // max int
    while(x > 21)
        x = 5 * (Rand5() - 1) + Rand5() // Rand25
    return x%7 + 1;
}
```

17

Randa RandbRandaRandb1a1bab ()RandaRandb

1. a > b2Randa2 = a * (Randa - 1) + Randa 1a2 a2 b Randa3 = a * (Randa2 - 1) + Randaak > bRandak , RandA
2. 1RandA(RandaRandak)A > b Randb

```
1       // A > b
2       int Randb(){
3         int x = ~(1<<31); // max int
4         while(x > b*(A/b)) // b*(A/b)AAb
5         x = RandA();
6         return x%b + 1;
7       }
```

RandaRandb Randab1a*b

- Randab = b * (Randa - 1) + Randb
- Randab = a * (Randb - 1) + Randa

ab cd

19.11 Design an algorithm to find all pairs of integers within an array which sum to a specified value.

Solution:

O(n) bitmap()sum-x O(n) bitmap 512345sum5 bitmapsum-x1432101.

O(nlogn) O(1) a-2 -1 0 3 5 6 7 9 13 14lowhigh a[low]+a[high] > suma[high] sum(a[low]sum) a[high]sum higha[low]+a[high] < suma[low] sum(a[high]sum) lowa[low]+a[high]sum lowhigh

```
1  #include <iostream>
2  #include <algorithm>
3  using namespace std;
4
5  void printPairSum(int a[], int n, int sum) {
6      if (a == NULL || n <= 0) return;
7
8      sort(a, a+n);
9      int low = 0, high = n-1;
10
11     while (low < high) {
12         if ( a[low] + a[high] > sum )
13             --high;
14         else if ( a[low] + a[high] < sum )
15             ++low;
16         else {
17             cout << a[low] << "␣" << a[high] << endl;
18             --high;
19             ++low;
20         }
21     }
22 }
23
24 int main() {
25     int n = 6, sum = 6;
26     int a[] = {1, 2, 3, 4, 5, 6};
27     printPairSum(a, n, sum);
28     return 0;
29 }
```

Chapter 20

Hard

20.1 Write a function that adds two numbers. You should not use + or any arithmetic operators.

Add+

Solution:

10759 + 674

YES """

1. 759 + 674323
2. 759 + 6741110
3. ()

1. (0,0),(1,1)0(1,0),(0,1)1
2. (1,1)
3. 0

```
int Add2(int a, int b){
    if(b == 0) return a;
    int sum = a ^ b; //
    int carry = (a & b) << 1; //
    return Add2(sum, carry); // sumcarry
}
```

```
int Add3(int a, int b){
    while(b != 0){
        int sum = a ^ b;
        int carry = (a & b) << 1;
        a = sum;
        b = carry;
    }
    return a;
}
```

aa[1] 1a[1]

a[1] a0xbfc86d98

- 0xbfc86d98 + sizeof(int) * 1 = 0xbfc86d9c
- 0xbfc86d9

```
int Add1(int a, int b){
    char *c = (char*)a;
    return (int)&c[b]; // c+sizeof(char)*b=a + b
}
```

acharcc[b]c[b] c + sizeof(char)*b = a + bbOK OK

c[b]b[c] a[5]5[a]

cC

20.2 Write a method to shuffle a deck of cards. It must be a perfect shuffle - in other words, each 52! permutations of the deck has to be equally likely. Assume that you are given a random number generator which is perfect.

52! 1/(52!)

Solution:

52515150 152! 1/(52!)

52cards

51234514 212354 124(2,3,1,5) 223

```
#include <iostream>
#include <cstdlib>
using namespace std;

void swap(int &a, int &b) { // swap
    int t = a;
    a = b;
    b = t;
}

void RandomShuffle(int a[], int n) {
    for (int i = 0; i < n; ++i) {
        int j = rand() % (n-i) + i; // in-1 // [i, n-1]
        swap(a[i], a[j]);
    }
}

int main() {
    srand( (unsigned)time(0) );
    int n = 9;
    int a[] = {1, 2, 3, 4, 5, 6, 7, 8, 9};
    RandomShuffle(a, n);
    for (int i = 0; i < 9; ++i)
        cout << a[i] << endl;
    return 0;
}
```

20.3 Write a method to randomly generate a set of m integers from an array of size n. Each element must have equal probability of being chosen.

nm

Solution:

1

1/n YES

- 1n1/n
- 2n-11/(n-1)1 n-1(n-1)/n(n-1)/n * 1/(n-1) = 1/n
- 3(n-1)/n * (n-2)/(n-1) * 1/(n-2) = 1/n

k1/n mm

```
#include <iostream>
#include <cstdlib>
using namespace std;

void swap(int &a, int &b) { // swap
    int t = a;
    a = b;
    b = t;
}

void PickMRandomly(int a[], int n, int m) {
    for (int i = 0; i < m; ++i) {
        int j = rand() % (n-i) + i;   // in-1
        swap(a[i], a[j]);
    }
}

int main() {
    srand( (unsigned)time(0) );
    int n = 9, m = 5;
    int a[] = {1, 2, 3, 4, 5, 6, 7, 8, 9};
    PickMRandomly(a, n, m);
    for (int i = 0; i < m; ++i)
        cout << a[i] << endl;
    return 0;
}
```

20.4 Write a method to count the number of 2s between 0 and n.

0n2

Solution:

0n2 2O(logn)n O(nlogn)

```
int Count2(int n) {
    int count = 0;
    while (n > 0) {
        if (n % 10 == 2) {
            ++count;
            n /= 10;
        }
    }
    return count;
}

int Count2s1(int n) {
    int count = 0;
    for (int i = 0; i <= n; ++i) {
        count += Count2(i);
    }
    return count;
}
```

n

5N=abcde20abcde 2 2

c0120130120132 200 299, 1200 1299, 2200 2299, , 11200 11299, 3200 2990111212200 12299 1201302 (12)x(100)=1200

c112113 11200 1129921200

- 22x

c212213200 299, 1200 1299, 2200 2299, , 11200 112991200212200 12213 14(+1)2 2

- 22x++1

c2123133200 299012 12200 122992 (+1)x

- 22(+1)x

3

```
int Count2s(int n) {
    int count = 0;
    int factor = 1;
    int low = 0, curr = 0, high = 0;

    while(n/factor != 0) {
        low = n - (n/factor)*factor; //
        curr = (n/factor) % 10;     //
        high = n / (factor*10);     //

        switch(curr) {
        case 0:
        case 1:
            count += high * factor;
            break;
        case 2:
            count += high * factor + low + 1;
            break;
        default:
            count += (high + 1) * factor;
            break;
        }
        factor *= 10;
    }
    return count;
}
```

0nii19 i00 0110123401020304 0

- iix
- iix++1
- ii(+1)x

```
int Countis(int n, int i) {
    if (i < 1 || i > 9) return -1; // i19 i should be in range [1, 9]

    int count = 0;
    int factor = 1;
    int low = 0, curr = 0, high = 0;

    while (n / factor != 0) {
        low = n - (n/factor)*factor; //
        curr = (n/factor) % 10;     //
        high = n / (factor*10);     //

        if (curr < i)
            count += high * factor;
        else if (curr == i)
            count += high * factor + low + 1;
        else
            count += (high + 1) * factor;
        factor *= 10;
```

```
20      }
21      return count;
22  }
```

20.5 You have a large text file containing words. Given any two words, find the shortest distance (in terms of number of words) between them in the file. Can you make the searching operation in O(1) time? What about the space complexity for your solution?

()O(1)

Solution:

isname1

- What is your name My name is Hawstein

isname nameis

isname isname

- What is your name My name is Hawstein
- 0 1 2 3 4 5 6 7
- p

pisnameis1name3 min=3-1=2pnamename5 is14minpisis6 name51minmin=1p isnameminO(n)O(1)

```
1   int ShortestDist(string text[], int n, string word1, string word2){
2       int min = kMaxInt / 2;
3       int pos1 = -min;
4       int pos2 = -min;
5
6       for(int pos=0; pos<n; ++pos){
7           if(text[pos] == word1){
8               pos1 = pos;
9               int dist = pos1 - pos2;
10              if(dist < min)
11                  min = dist;
12          }
13          else if(text[pos] == word2){
14              pos2 = pos;
15              int dist = pos2 - pos1;
16              if(dist < min)
17                  min = dist;
18          }
19      }
20      return min;
21  }
```

O(1)

- What is your name My name is Hawstein
- 0 1 2 3 4 5 6 7

()

-
- What: 3 0
- is: 7 1, 6
- your: 13 2
- name: 14 3, 5
- My: 25 4
- Hawstein: 27 7

O(1)

O(1)O(1) isname(¡)n O(nlogn)

keyvalue value O(1)O(n2)

isnameisnamenameisnameis () nameisi<nisname

()

- (isWhat) 8 1
- ... ...
- (isname) 12 1
- ... ...
- (isMy) 33 2
- ... ...

isnameisname isname12

isnameisMy

- 8 (isWhat,1)
- ... ...
- 12 (isname,1) -> (isMy,2)
- ... ...

str12 strisname1 strisMy2

isinamej d[i][j] dd[i][j] i<j

```
#include <iostream>
using namespace std;

const int kMaxInt = ~(1<<31);

int ShortestDist(string text[], int n, string word1, string word2) {
    int min = kMaxInt / 2;
    int pos1 = -min;
    int pos2 = -min;

    for (int pos = 0; pos < n; ++pos) {
        if (text[pos] == word1) {
            pos1 = pos;
            int dist = pos1 - pos2;
            if (dist < min)
                min = dist;
        }
        if (text[pos] == word2) {
            pos2 = pos;
            int dist = pos2 - pos1;
            if (dist < min)
                min = dist;
        }
    }
    return min;
}

int main() {
    string text[] = {"What", "is", "your", "name", "My", "name", "is", "Hawstein"};
    int len = 8;
    cout << ShortestDist(text, len, "is", "name") << endl;
    return 0;
}
```

20.6 Describe an algorithm to find the largest 1 million numbers in 1 billion numbers. Assume that the computer memory can hold all one billion numbers.

10110

Solution:

101 10010

1O(nlogn)

11 10 1nkO(nlogk) kn

()kk nk ()

k kpartitionpivot (1)pivotpivot pivotpartition pivotpivot pivotk-1k kkpivot k-1partitionpivot k-1partition O(n)

20.7 Write a program to find the longest word made of other words.

Solution:

```
string arr[] = {"test", "tester", "testertest", "testing", "apple", "seattle", "banana",
"batting", "ngcat", "batti","bat", "testingtester", "testbattingcat"};
```

OK

bat testbattingcat testtestbattingcat battingcat testbattingcat

1. ()
2. ss true

testbattingcattte testestbattingcatb babattingcat battingcat batbattbatti ngcatnngngcngcangcat

```
#include <cstring>

const int kWordSize = 26 + 5;
const int kNodeSize = 1200 + 5;
const int kHashSize = 10001;

struct Node{
    char word[kWordSize];
    Node *next;
};
Node node[kNodeSize + 1];
Node* head[kHashSize + 1];

class Hash{
public:
    Hash();
    unsigned int hash(const char* str);
    void insert(const char* str);
    bool find(const char* str);
private:
    unsigned int seed;
    unsigned int size;
};

Hash::Hash():seed(131),size(0){
    memset(head, 0, sizeof(head));
}

unsigned int Hash::hash(const char* str){
    unsigned int hash = 0;
    while(*str++)
        hash = hash * seed + (*str);
    return (hash & 0x7FFFFFFF) % kHashSize;
}

void Hash::insert(const char* str){
    unsigned int id = hash(str);
    char *dst = (char*)node[size].word;
    while(*dst++ = *str++);
```

```
40      node[size].next = head[id];
41      head[id] = &node[size];
42      ++size;
43  }
44
45  bool Hash::find(const char* str){
46      unsigned int id = hash(str);
47      for(Node* p=head[id]; p ; p=p->next){
48          char *dst = (char*)p->word;
49          int i = 0;
50          while(*(str+i) && *(dst+i)==*(str+i))
51              ++i;
52          if(*(str+i)=='\0' && *(dst+i)=='\0')
53              return true;
54      }
55      return false;
56  }
```

```
1   #include <iostream>
2   #include <algorithm>
3   #include "hash.h"
4   using namespace std;
5
6   Hash hash;
7
8   inline bool cmp (string s1, string s2) { //
9       return s2.length() < s1.length();
10  }
11
12  bool MakeOfWords(string word, int length) {
13      int len = word.length();
14      if (len == 0) return true;
15
16      for (int i = 0; i <= len; ++i) {
17          if (i == length) return false; //
18          string str = word.substr(0, i);
19          if ( hash.find((char*)&str[0]) ) {
20              if ( MakeOfWords(word.substr(i), length) )
21                  return true;
22          }
23      }
24      return false;
25  }
26
27  void PrintLongestWord(string word[], int n) {
28      for (int i = 0; i < n; ++i)
29          hash.insert( (char*)&word[i][0] );
30      sort(word, word+n, cmp);
31
32      for (int i = 0; i < n; ++i) {
33          if ( MakeOfWords(word[i], word[i].length()) ) {
34              cout << "Longest Word: " << word[i] << endl;
35              return;
36          }
37      }
38  }
39
40  int main() {
41      string arr[] = {"test", "tester", "testertest", "testing",
```

```
42                      "apple", "seattle", "banana", "batting", "ngcat",
43                      "batti", "bat", "testingtester", "testbattingcat"};
44      int len = 13;
45      PrintLongestWord(arr, len);
46      return 0;
47  }
```

O(1)O(nlogn) O(d)(d) nO(nd)O(nlogn + nd) nO(nd)nO(nlogn)

```
1   if(i == length) return false;//
```

()true
abcdefg2 abcdefgabcdefg ()
O(n)n dO(nd*n)=O(dn2) O(logn)O(dnlogn)

20.8 Given a string s and an array of smaller strings T, design a method to search s for each small string in T.
ST(TS) ST
Solution:

STSm nkKMP(BM) O(m+n)O(k(m+n)) mn (k) KMPBM
TrieACWM Trie
Trie 2610
Trie
abc, abcd, abd, b, bcd, efg, hig, Trie ()
http://hawstein.com/posts/20.8.html
ST() Trie
S = abcd"

- abcd
- bcd
- cd
- d

tStSt = bc bcdt = ccd
STrie(Trie) Trietn O(n)tS
BANANASTrie
TrienO(n) STrieO(m2) (mS)O(m2)

```
1   #include <iostream>
2   #include <cstring>
3   using namespace std;
4
5   class Trie {
6   public:
7       static const int MAX_N = 100 * 100; // 100
8       static const int CLD_NUM = 26;      // (26)
9       int size;       //
10      int trie[MAX_N][CLD_NUM];
11
12      Trie(const char* s);
13      void insert(const char* s);
14      bool find(const char* s);
15  };
16
17  Trie::Trie(const char* s) {
18      memset(trie[0], -1, sizeof(trie[0]));
19      size = 1;
20      while(*s) {
```

```
21          insert(s);
22          ++s;
23      }
24  }
25
26  void Trie::insert(const char* s) {
27      int p = 0;
28      while (*s) {
29          int i = *s - 'a';
30          if ( -1 == trie[p][i] ) {
31              memset( trie[size], -1, sizeof(trie[size]) );
32              trie[p][i] = size++;
33          }
34          p = trie[p][i];
35          ++s;
36      }
37  }
38
39  bool Trie::find(const char* s) {
40      int p = 0;
41      while(*s) {
42          int i = *s - 'a';
43          if ( -1 == trie[p][i] )
44              return false;
45          p = trie[p][i];
46          ++s;
47      }
48      return true;
49  }
50
51  int main() {
52      Trie tree("mississippi");
53      string patt[] = {"is", "sip", "hi", "sis", "mississippa"};
54      int n = 5;
55      for (int i = 0; i < 5; ++i)
56          cout << tree.find((char*)&patt[i][0]) << endl;
57      return 0;
58  }
```

O(m2 + kn) AC
ACO(m+kn+z) mSknzS ACTrie m21
ACTrieTrie SS
AC Set Matching and Aho-Corasick Algorithm (http://www.cs.uku.fi/~kilpelai/BSA05/lectures/slides04.pdf) (http://www.cs.uku.fi/

AC

```
1  #include <iostream>
2  #include <queue>
3  #include <cstring>
4  using namespace std;
5
6  class ACAutomation {
7  public:
8      static const int MAX_N = 1000 * 50;
9      static const int CLD_NUM = 26;
10
11     int size;
12     int fail[MAX_N];
13     int tag[MAX_N];
14     int trie[MAX_N][CLD_NUM];
```

```
15
16      void reset();
17      void insert(const char* s);
18      void construct();
19      int query(const char* s);
20  };
21
22  void ACAutomation::reset() {
23      memset( trie[0], -1, sizeof(trie[0]) );
24      tag[0] = 0;
25      size = 1;
26  }
27
28  void ACAutomation::insert(const char* s) {
29      int p = 0;
30      while (*s) {
31          int i = *s - 'a';
32          if ( -1 == trie[p][i] ) {
33              memset( trie[size], -1, sizeof(trie[size]) );
34              tag[size] = 0;
35              trie[p][i] = size++;
36          }
37          p = trie[p][i];
38          ++s;
39      }
40      ++tag[p];
41  }
42
43  void ACAutomation::construct() {
44      queue<int> q;
45      fail[0] = 0;
46      for (int i = 0; i < CLD_NUM; ++i) {
47          if ( trie[0][i] != -1 ) {
48              fail[ trie[0][i] ] = 0;
49              q.push( trie[0][i] );
50          } else
51              trie[0][i] = 0;
52      }
53      while ( !q.empty() ) {
54          int u = q.front();
55          q.pop();
56          for (int i = 0; i < CLD_NUM; ++i) {
57              int &v = trie[u][i];
58              if (v != -1) {
59                  q.push(v);
60                  fail[v] = trie[ fail[u] ][i];
61                  // tag[u] += tag[ fail[u] ];
62              } else
63                  v = trie[ fail[u] ][i];
64          }
65      }
66  }
67
68  //
69  int ACAutomation::query(const char* s) {
70      int p = 0, cnt = 0;
71      while (*s) {
72          int i = *s - 'a';
73          while ( trie[p][i] == -1 && p != 0 ) //fail
```

```
74              p = fail[p];
75          p = trie[p][i];
76          p = p==-1 ? 0 : p;
77          int t = p;
78          while (t != 0 && tag[t] != -1) {
79              cnt += tag[t];
80              tag[t] = -1;
81              t = fail[t]; //
82          }
83          ++s;
84      }
85      return cnt;
86  }
87
88  int main() {
89      ACAutomation ac;
90      ac.reset();
91      string patt[] = {"is", "sip", "is", "sis", "mississipp"};
92      int n = 5;
93      for (int i = 0; i < n; ++i)
94          ac.insert( (char*)&patt[i][0] );
95      ac.construct();
96      cout << ac.query("mississippi") << endl;
97      return 0;
98  }
```

Ukkonen 3 Gusfield Algorithms on strings, trees, and sequences

- () Fast String Searching With Suffix Trees: (http://marknelson.us/1996/08/01/suffix-trees/)
- ()-Ukkonen: http://blog.163.com/lazy_p/blog/static/1351072162010813947681 6/
- stackoverflow Ukkonens suffix tree algorithm in plain English: (http://stackoverflow.com/questions/9452701/ukkonens-suffix-tree-algorithm-in-plain-english)
- : (http://www.ibaiyang.org/2013/01/06/suffix-tree-introduction/)

20.9 Numbers are randomly generated and passed to a method. Write a program to find and maintain the median value as new values are generated.

Solution:

A O(1)

- 1 3 5 1 3 5 7
- 0 1 2 0 1 2 3
- A[n/2] (A[(n-1)/2] + A[n/2])/2

O(n) 1

()O(logn) O(nlogn)

O(logn) n n () n 2 n1 1 1

O(logn)O(1)

```
1  #include <iostream>
2  #include <algorithm>
3  #include <queue>
4  #include <cstdlib>
5  using namespace std;
6
7  class Median {
8  private:
9      priority_queue< int, vector<int>, less<int> > max_heap;    //
10     priority_queue< int, vector<int>, greater<int> > min_heap;//
11 public:
```

```
12      void Insert(int v);
13      int GetValue();
14  };
15
16  void Median::Insert(int v) {
17      if ( max_heap.empty() && min_heap.empty() )
18          max_heap.push(v);
19      else if ( !max_heap.empty() && min_heap.empty() )
20          max_heap.push(v);
21      else if ( max_heap.empty() && !min_heap.empty() )
22          min_heap.push(v);
23      else {
24          if (v < max_heap.top() )
25              max_heap.push(v);
26          else
27              min_heap.push(v);
28      }
29
30      // 1
31      // max_heap.size()-min_heap.size()>1
32      // sizeunsignedfalse
33      // true
34      while( max_heap.size() > min_heap.size() + 1) {
35          int data = max_heap.top();
36          min_heap.push(data);
37          max_heap.pop();
38      }
39      while( min_heap.size() > max_heap.size() + 1) {
40          int data = min_heap.top();
41          max_heap.push(data);
42          min_heap.pop();
43      }
44  }
45
46  int Median::GetValue() {   //intfloat
47      if (max_heap.empty() && min_heap.empty())
48          return (1<<31);    //int
49
50      if (max_heap.size() == min_heap.size())
51          return (max_heap.top() + min_heap.top())/2;
52      else if (max_heap.size() < min_heap.size())
53          return min_heap.top();
54      else
55          return max_heap.top();
56  }
57
58  int main() {
59      srand( (unsigned)time(0) );
60      Median md;
61      vector<int> vi;
62      int num = rand() % 30;   //30
63      for (int i = 0; i < num; ++i) {
64          int data = rand() % 100; //100
65          vi.push_back(data);
66          md.Insert(data);
67      }
68      sort(vi.begin(), vi.end());
69      for (int i = 0; i < num; ++i)
70          cout << vi.at(i) << "␣";          //
```

```
71      cout << endl << md.GetValue() << endl;   //
72      return 0;
73  }
```

20.10 Given two words of equal length that are in a dictionary, write a method to transform one word into another word by changing only one letter at a time. The new word you get in each step must be in the dictionary.
EXAMPLE
Input: DAMP, LIKE
Output: DAMP -> LAMP -> LIMP -> LIME -> LIKE

20.11 Imagine you have a square matrix, where each cell is filled with either black or white. Design an algorithm to find the maximum subsquare such that all four borders are filled with black pixels.

Solution:
()

```
1  #include <iostream>
2  #include <cstdio>
3  using namespace std;
4
5  const int MAX_N = 100;
6  int matrix[MAX_N][MAX_N];
7
8  struct SubSquare {
9      int row, col, size;
10 };
11
12 inline int max(int a, int b) {
13     return a > b ? a : b;
14 }
15
16 bool IsSquare(int row, int col, int size) {
17     for (int i = 0; i < size; ++i){
18         if (matrix[row][col+i] == 1)    // 10 1: while; 0: black
19             return false;
20         if (matrix[row+size-1][col+i] == 1)
21             return false;
22         if (matrix[row+i][col] == 1)
23             return false;
24         if (matrix[row+i][col+size-1] == 1)
25             return false;
26     }
27     return true;
28 }
29
30 SubSquare FindSubSquare(int n) {
31     int max_size = 0;    // // max length for one size
32     int col = 0;
33     SubSquare sq;
34     while (n-col > max_size) {
35         for (int row = 0; row < n; ++row) {
36             int size = n - max(row, col);
37             while (size > max_size) {
38                 if ( IsSquare(row, col, size) ) {
39                     max_size = size;
40                     sq.row = row;
41                     sq.col = col;
42                     sq.size = size;
43                     break;
```

```
44                      }
45                      --size;
46                  }
47              }
48              ++col;
49          }
50          return sq;
51      }
52
53      int main() {
54          freopen("ch20.11.in", "r", stdin);
55          int n;
56          cin >> n;
57          for (int i = 0; i < n; ++i)
58              for (int j = 0; j < n; ++j)
59                  cin >> matrix[i][j];
60          SubSquare sq = FindSubSquare(n);
61          cout << "top: " << sq.row << endl;
62          cout << "bottom: " << sq.row + sq.size -1 << endl;
63          cout << "left: " << sq.col << endl;
64          cout << "right: " << sq.col + sq.size -1 << endl;
65          fclose(stdin);
66          return 0;
67      }
```

20.12 Given an NxN matrix of positive and negative integers, write code to find the submatrix with the largest possible sum.

NxN(0)

Solution:

: O(n6)

C(n, 2)*C(n, 2)() O(n4)O(n2) O(n6)

: O(n4)

O(n2) O(1)

pp[i][j](1, 1)(1) (i, j)(x1, x2, y1, y2) (D)

```
1   sum(D) = p[y2][x2] - p[y2][x1-1] - p[y1-1][x2] + p[y1-1][x1-1]
```

O(1)

p[i][j]

```
1   p[i][j] = p[i-1][j] + p[i][j-1] - p[i-1][j-1] + A[i][j]
```

A[i][j](i, j)O(n2)

: O(n3)

O(n)

k l i... ... ... j... ... ... ij

k l ... ... O(n)kl ijkl

ijO(n2)O(n) O(n3)k ijO(1)

```
1   sum(i,j,k) = p[j][k] - p[j][k-1] - p[i-1][k] + p[i-1][k-1]
```

```
1   #include <iostream>
2   #include <cstdio>
3   using namespace std;
4
```

```
5  const int MAX_N = 100;
6  int p[MAX_N][MAX_N], A[MAX_N][MAX_N];
7
8  void PreCompute(int n){
9      for (int i = 0; i <= n; ++i)
10         p[0][i] = p[i][0] = 0;
11     for (int i = 1; i <= n; ++i)
12         for (int j = 1; j <= n; ++j)
13             p[i][j] = p[i-1][j] + p[i][j-1] - p[i-1][j-1] + A[i][j];
14 }
15
16 int MaxSum(int n) {
17     int max_sum = 1 << 31; // Min Int
18     for (int i = 1; i <= n; ++i)
19         for (int j = 1; j <= n; ++j) {
20             int curr_sum = 0;
21             for (int k = 1; k <= n; ++k) {
22                 int val = p[j][k] - p[j][k-1] - p[i-1][k] + p[i-1][k-1];
23                 if (curr_sum <= 0)
24                     curr_sum = val;
25                 else
26                     curr_sum += val;
27                 if (curr_sum > max_sum)
28                     max_sum = curr_sum;
29             }
30         }
31     return max_sum;
32 }
33
34 int main() {
35     freopen("ch20.12.in", "r", stdin);
36     int n;
37     cin >> n;
38     for (int i = 1; i <= n; ++i)    //1
39         for (int j = 1; j <= n; ++j)
40             cin >> A[i][j];
41     PreCompute(n);
42     cout << MaxSum(n) << endl;
43     fclose(stdin);
44     return 0;
45 }
```

20.13 Given a dictionary of millions of words, give an algorithm to find the largest possible rectangle of letters such that every row forms a word (reading left to right) and every column forms a word (reading top to bottom).