unixCommands

```
$ls –l –a -1 | wc -l
$cd dir (~ .. .) $pwd
$mkdir –p dir  $rmdir dir  $rm –rf dir
```

```
$chmod a+x abc.py
$chmod 777 abc.py
# r:100, w:010, x:001  drwxrwxrwx
```

```
$cp file1 file2    # copy file1 into file2;   -r: including sub-directories
$mv [file dir] [dir dir2] [file1 file2] # rename or move accordingly
$rm file  $touch file  $cat file
$sed 's/unix/linux/g' test.txt
```

```
$find . –name "hello.*" –type [f|d]
$grep –E 'REGEX' filename   -c count
. single char
^ beginning
$ end;   ^...$ 3 chars
```

```
X? 0-1     X* 0+     X+ 1+
[A-Za-z]*    .*  >=0 chars  (ab)+
```

```
[1-5][A-Z][A-Za-z]
\ escape
(ab){n} {n,m} {n,}
```

```
sort –r –n –k 1 –t, -u file    uniq –c file   wc –w –b –l file
cut –d ' ' –f 1,3 "$1"   head/tail –n 5   # get first/last n lines
diff file1 file2     spell file           # incorrect words
${get[0]} # first word
```

shellScript

```
#!/bin/bash

echo "hi$0"    # w/    \n
echo –n 'hi'   # w/o   \n

a="cat"    # setting value no $
echo $a    # getting value need $
read name  # user stdin
d="`ls -1`"   # assign cmd stdout

${#a}            # strlen
${a:pos:len}     # closed, 0 based
${a/$from/$to}   # replace 1st occ.

let "a=$a+1"     # arithmetic
```

```
if [ condition1 ]
then
    statement
elif [ condition2 ]
then
    statement
else
    statement
fi

| < 1> 2> 1>> 2>>
```

```
l=`ls -1`
cnt=wc –l $l

for item in $list
do
  # each word/line
done

for ((i=0; i<=100;
i=i+3 ));
do echo $i;
done
```

```
echo $#      # number of arguments
echo $0      # filename
echo $1      # frst argument
echo ${10}   # 10th argument

cp file123 fileabc 1>/dev/null 2>&1
echo "$0: failed" >&2
```

```
[ "$string" ]
[ "$str1" ?? "$str2" ] #  == != \> \<

[ -? $file ]
e: exists     -s: nonempty
f: file       -r/w/x: have that perm
d: directory

[ $a -?? $b ]   #  -eq/ne/lt/le/gt/ge
```

Makefile filename is Makefile g++ a.cpp –o a.o

```
census.o: census.cpp BigInteger.h
     g++ -c census.cpp

BigInteger.o: BigInteger.h BigInteger.cpp
     g++ -c BigInteger.cpp

census: census.o BigInteger.o
     g++ census.o BigInteger.o -o census

run: census # if no dependency, still go to next line
     ./census > output.txt
     echo "The following is the output of census:\n"`cat output.txt`

.PHONY: run    # $ make run

#   $@  target    $^  dependency list   $<  leftmost item in dependency list
```

cppClassDeclarationOperatorOverload

BST

```
// BigInteger.h
#include <string>
using namespace std;

class BigInteger{
    public:
        void setNumber( string );
        string getNumber() const;
        BigInteger( string );
        BigInteger();
    private:
        char sign;
        int length;
        int value[100];
    friend BigInteger operator+( const BigInteger & a, const BigInteger & b );
    friend ostream & operator<<(ostream & cout, const BigInteger & b );
    friend istream & operator>>(ostream & cin, BigInteger & b );
};

BigInteger operator+( const BigInteger & a, const BigInteger & b );
ostream & operator<<(ostream & cout, const BigInteger & b );
istream & operator>>(istream & cin, BigInteger & b );
```

```
#include <stdio.h>
#include <stdlib.h>
#define max(a,b) (((a)>(b))?(a):(b))

struct Node{
       int key;
       struct Node *left;
       struct Node *right;
};

typedef struct Node Node;
```

```
// BigInteger.cpp
#include <iostream>
#include <string>
#include "BigInteger.h"
using namespace std;

void BigInteger::setNumber( string num ){  sign='+', length = num.length(); }

string BigInteger::getNumber() const{ // const function: not modify member var
    string s; for(int x:value) s=s+(char)(x+'0');  return s;
}

BigInteger::BigInteger(string s) {       setNumber(s);  }
BigInteger::BigInteger() {  setNumber("0");  } // default constructor

BigInteger operator+( const BigInteger& a, const BigInteger& b ){
    BigInteger c;    /* implementation. can return b+a to swap inputs */
    return c;
}

ostream & operator<< (ostream &cout, const BigInteger &b){
    string s = b.getNumber();
    cout << s;
    return cout;
}

istream & operator>> (istream &cin, BigInteger &b){
    string s; cin >> s;
    b.setNumber(s);
    return cin;
}
```

```
int main(){
  Node * root = NULL;
  root = Insert(root,x);
  Node * temp = Find(root,10000);
  if (temp==NULL) printf("Not found\n");
}
```

```
// main.cpp
#include <iostream>
#include "BigInteger.h"
using namespace std;
int main(){
    BigInteger b1, b2, b3;
    cin >> b1 >> b2;
    b3=b1+b2;
    BigInteger c,d("123");
    cout << b1+b2 << endl;
    return 0;
}
```

BST

```
Node * Insert (Node * cur, int key){
  if (cur == NULL){
       Node * temp;
       temp = (Node *) malloc(sizeof(Node));
       temp->key = key;
       temp->left = temp->right = NULL;
       return temp;
  }
  if (key > (cur -> key))
       cur -> right = Insert(cur -> right, key);
  else cur -> left = Insert(cur -> left, key);
  return cur;
}
```

```
Node * Find(Node * cur, int key){
  if (cur == NULL) return NULL;
  if (key > cur -> key) return Find(cur -> right, key);
  else if (key < cur -> key) return Find(cur -> left, key);
       return cur;
}

Node *FindMin(Node * cur){
  if (cur == NULL) return NULL;
  if (cur -> left == NULL) return cur;
  return FindMin(cur -> left);
}

int get_h(Node * cur){
  if (cur == NULL) return 0;
  return 1+max(get_h(cur -> left), get_h(cur -> right));
}
```

stlContainersIteratorsAlgorithmsTemplate

```
vector<int> a;  vector<int> a(100);  vector<int> a(n, 100); // l.compare < <= == !=
vector<int> a = {2,4,2,5,7};    vector<vector<int> > a;
vector<vector<int>> a(n, vector<int> (m)); // 2D vector size (n,m)
a.push_back(123); a.pop_back(); int sz = a.size(); int x = a[2];
for (vector<int>::iterator it = a.begin(); it! = a.end(); it++)  sum+=*it;

a.erase(a.begin()+2); // 3rd only;  a.erase(a.begin(),a.begin()+2); on range
a.insert(a.begin()+2, 200);   // insert element before that pointed by Itr

void print(const vector<int>& v){
    for (vector<int>::const_iterator it; it!=v.end(); it++) cout << *it << endl;
}
```

```
list<T>  // Bidirectional linked list; // O(1) insertion/erase, No random access
list<int> l{10,20}; l.push_back(30);  l.push_front(0);
cout << l.front() << " " << l.back() << endl; // 0  30
list<int>::iterator it = l.begin();   // >0  10  20  30
++it;                    //  0 >10  20  30
it = l.erase (it);       //  0 >20  30
++it;                    //  0  20  >30
it = l.insert(it,25);    //  0  20  >25  30
l.insert(l.end(), 40);   //  0  20  >25  30  40
cout << *it << endl;     //  25
l.sort();  l.unique();  l.reverse();
```

```
map<K, V>  // balanced binary search tree; // O(logn) insertion/erase, unique keys
V& operator[](const K& k); // retrieve element and assign
m.count(123); // 0: not exist; 1: exist, avoid creating new node
// for user-defined objects, define   bool operator<(const Rec &a, const Rec &b);
for (map<K,V>::iterator it = m.begin(); it != m.end(); it++); // sorted in K
```

```
sort(v.begin(), v.end()); sort(v.begin(), v.begin()+5); l.sort();
bool cmp(int,int); sort(v.begin(), v.end(), cmp); bool cmp(int a,int b){return b<a;}

bool binary_search(FwdIt first, FwdIt last, const T& target);
FwdIt lower_bound(FwdIt first, FwdIt last, const T& target); // 1st pos >= x
FwdIt upper_bound(FwdIt first, FwdIt last, const T& target); // 1st pos >  x
// work on list and map as well, but take O(n) time
// n = it_lo – it_up;   n is number of occurrence
```

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <cstdlib>
#include <ctime>
using namespace std;

template <class T>
class MyCollection{
          vector<T> data;
     public:
          void Add(T const &);
          T & Draw();
   template <class U>
   friend ostream & operator<<
        (ostream& cout,
         const MyCollection<U>& q);
};
```

```
template <class U>
ostream & operator<<( ostream & cout,
                const MyCollection<U>& q){
    typename vector<U>::const_iterator itr;
    for ( itr= q.data.begin() ;
          itr != q.data.end() ; itr++ )
        cout << " " << *itr <<endl;
    return cout;
}

template <class T>
void MyCollection<T> ::Add(T const &d){
     data.push_back(d);
}
```

```
template<class T>
T maximum (const T& a, const T& b){
    return a < b ? b : a;
}
```

```
maximum(10,20);             // 20
maximum(10,20.5);           // 20.5

maximum<int>(10,20.5);      // 20
maximum<double>(10,20.5);   // 20.5
```

linkedList

```
void printList (Node* head){ Node* cur=head; while(cur!=NULL) cur=cur->next; }
void insert (Node **head_ref) {Node* temp = (Node*) malloc(sizeof(Node));
   temp->next=NULL, temp->data=data; *head_ref=temp;} // node: int data, Node* next
```

c99

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

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

void toLower(char a[]){
  int len = strlen(a), i;
  for (i=0; i<len; i++) a[i] = a[i]|32;
}

struct Student{
  char name[20];
  int uid;
};
typedef struct Student Student;
```

```
int main(){
  int a[]={1,2,3,4};
  char name[] = "alan";
  printf("%s",name);
  int x; char a[100];
  scanf("%d %s",&x,a);  // %d %f %c %s %g

  struct Student *a;
  a = (Student*) malloc (size *
                  sizeof(Student));

  free(a);
  return 0;
}
```

```
Strcpy(str1,str2): copy str2 into str1
strcat(str1,str2): str1 = str1 + str2
strlen(str): length, remember \0 ending
strcmp(str1,str2): -1/0/1  < == >
```

Python: chmod +x abc.py ./abc.py

```
#!/usr/bin/python
import math
a = 5 # a is int, no ++/--
b = 3.0 # b is float
c = "10" # c is string
print a/2 # 2
print b/2 # 1.5
print a/b # 1.6666666667
print round(a/b,3) # 1.667
print math.ceil(2/3.0)
```

```
print "hello"    # with \n
print "hello",   # witout \n
print """line1
line2
line3""" # with linebreaks
```

```
s = "abc" + "def"
print s # abcdef, with \n
print s[3]    # d
print s[1:5]  # bcde
print s[:4]   # abcd
print s[1:]   # bcdef
print len(s)  # 6
```

```
s = input("name: ") # return str
print "Hello " + s

n = int(input()) # casting
pow(2,3)
ord('A') # ascii A   chr(65) # str('A')
```

```
a = "12" + "3"
b = 3
print int(a) + b # 126
print a + str(b) # 1233
print \"  # ", escape char
```

flowOfControl

```
if condition: # and, or, not
    statement
elif condition:
    statement
    statement
else:
    statement
```

```
while condition: # True, False
    statement
    break
    continue

for i in list: # [2,3,7,9]   range(0,n) open  range(n)
    statement
```

fileIO

```
#!/usr/bin/python
with open ("number.txt", "r") as f:
    count = f.read() # str
    n = int(count)
f.close()

with open("number.txt", "w") as f:
    f.write(str(n+1)) # str
f.close()

with open("list.txt", "w") as f:
    for line in f:
        words = line.split(' ');
        for word in words:
            print (word)
```

```
r:  r only
r+: r/w
w:  w only, overwrite/create new
w+: r/w, overwrite/create new
a:  append/create new
a+: r/append/create new
```

```
words = line.split(',')
words[i] = foo
words = []       # empty list
words.append(bar)  len(words)
words.sort()  words.reverse()

for c in word:
    # for each character
```