Cheatography

COMP2123 Cheat Sheet

by Jack84 via cheatography.com/72157/cs/18281/

Bashshell Basic Commands

`gedit example.txt &`	create a new empty file

1.1 Bashshell Directory Manipulation

`pwd`	Name of the current directory
`cd dir`	change the directory to `cd`
`cd ~`	Change to home directory
`cd ..`	Change to parent directory
`cd .`	Change to current directory
`mkdir dir`	Creates a new directory
`rmdir dir`	removes a directory
`rm -rf dir`	Removes non-empty directory dir and all of its subdirectories and files
`mv dir1 dir2`	Rename dir1 to dir2

1.2 Bashshell File Manipulation

`gedit hello.cpp &`	Starts the gedit program to edit file hello.cpp
`g++ hello.cpp -o hello.o`	compile hello.cpp to create executable a.o
`./a.o`	invoke `a.o`
`cp file1 file2`	Copy file1 into file2
`cp -r dir1 dir2`	Copy dir1 (including its subdirectories) to dir2

1.2 Bashshell File Manipulation (cont)

`mv file dir`	moves `file` to `dir`
`rm file`	remove file
`rm dir`	remove all files and directories in `dir`
`touch file`	Create an empty file called `file`
`cat file`	Display the content of `file`

File Security and Permission

`ls -lt` print in long form with security level indication

`chmod [who] [operator] [permissions] filename`

[who] `u` = user; `g` = group; `o` = other; `a` = all

[operator] `+` = add; `-` = remove; `=` set

[permission] `r` = read; `w` = write; `x` = execute

1.3 Bash Shell Searching

`find . -name "hello.txt" -type f`	Search by name for a file
`find . -name "hello.*" -type f`	Search any file type with the name "hello"
`find . -name "hello" -type d`	Search for a directory called "hello"
`find . -name "*.c"`	find .c files

1.3 Bash Shell Searching (cont)

`find . -name "*.[ch]"	find .c and .h files
`grep '^apple' example`	match beginning of a line
`grep 'apple$' example`	match the end of a line
`grep '^apple$' example`	match the exact contents of a line
`grep 'p?' example`	match zero or one occurrence
`grep 'p+' example`	match one or more occurrences
`grep 'p*' example`	match zero or more occurrences
`grep 'p.' example`	match a single character "p"
`'[12345]'` or `'[1-5]'`	match any character enclosed by []
`(ab){3}`	3 occurrences of "ab"
`(ab){1,3}`	1 to 3 occurrences of "ab"
`(ab){3,}`	3 or more occurrences of "ab"
`grep -c "UNIX" bar.txt`	no. of occurrence for "UNIX"

1.3 Bash Shell Searching (cont)

`sed s/UNIX/Unix/g bar.txt`	replace all "UNIX" to "Unix"

Note that (ab) can also be other valid patterns

1.4 Useful Bash Shell Commands

`wc -c file`	return no. of bytes
`wc -l file`	returns no. of lines
`wc -w file`	returns no. of words
`sort file`	sort file in alphabetical order and output the sorted texts
`sort -n file`	numerical sort
`sort -n -r file`	-r for reverse sort
`sort -k3 -n filename`	sort by field no. 3
`sort -t, -k3 -n filename`	sort by field no.3 but the delimiter is comma
`cut -d ' ' -f 1,3 example1`	return 1st and 3rd columns where the delimiter is a space
`uniq file`	remove adjacent duplicate lines
`diff fileA fileB`	how to transform fileA to fileB
`0a1`	add the line1 of fileB after line0 of fileA

1.4 Useful Bash Shell Commands (cont)

`2,3C3`	change line2,3 of fileA to line3 of fileB
`spell file`	display all incorrect words in file
`su`	change to super user mode
`yum install [program]`	install program

1.5 Standard I/O and pipe

`wc data.txt 1> result.txt`	send standard output to file
`wc data.txt > result.txt`	same as above
`[command] >> file`	append result of [command] to the file
`[command] 2> file`	send standard error to file
`./add.o < input.txt > output.txt`	add.o takes input from input.txt and output to output.txt

```
ls -l | grep "Jan 25"
```

C++ friend

```
// --- In .h file ---
class BigInteger{
  public:
    void setNumber( string );
    string getNumber();
  private:
    char sign;
    int length;
    int value[100];
  friend BigInteger add(BigInteger
a, BigInteger b);
};
BigInteger add(BigInteger a,
BigInteger b);
// add declaration of friend
function AFTER the declaration fo
the big integer class
// --- in .cpp file --- //
BigInteger add(BigInteger a,
BigInteger {
  // implement the function
}
```

C++ Constructor

```
class Point{
private:
  int x, y;
public:
    Point() { // default
      a = 10; b=20
```

C++ Constructor (cont)

```
    }
    Point(int x1, int y1){ //
parameterised
      x = x1; y = y1;
    }
}
int main(){
  Point p; // default
  Point q(10, 20); //
parameterised
}
```

Constructor is a member function that shares the same name as the class

C++ const keyword

```
BigInteger add(const BigInteger &
a, const BigInteger & b)
{
  // Cannot modify any of the
parameters
}
string
BigInteger::setNumber(string
number) const {
  // setNumber is a read-only
function.
  sign = "-"; // error because sign
is a member variable of BigInteger
}
```

C user input

```
int a; float b;
scanf("%d%f", &a, &b);
printf("%g", a*b);
```

2.1 Shell Script basics

echo -n "hello world"	print without \n
read name	read user input and store it in a var called "name"
a=apple	no quote
a='apple pie'	single quote (strings)
a="$a\$"	can handle special characters
$	variable substitution
\	escape special characters
`` `cat hello.txt` ``	enclose bash commands

```
a="`wc -l file | cut -d\"\" -f1`"
```

```
echo "there are $a lines in file"
```

2.2.1 Shell Script - Using Strings

${#a}	length of string
${a:pos:len}	substring (assume index 0)
${a/from/to}	change part of string

```
a="Apple pie"; from="pie";
to="juice";
```

2.2.4 Shell Scripting - Variable as numbers

let "a=$a+1"	increment a by 1

2.3 Shell Scripting - Control Flow

```
# --- if-else statements --- #
if [ condition ]
then
  echo "Action 1"
elif [ condition2 ]
then
  echo "Action2"
else
  echo "Action neither"
fi
#example
#!/bin/bash
echo "Do you want to remove all
.cpp files (Y/N)"
read ans
if [ "$ans" == "Y" ]
then
  rm -rf *.cpp
  echo "All .cpp files are
removed"
fi
# --- for-in loop --- #
#!/bin/bash
list="1 2 3 4 5"
for i in $list
```

2.3 Shell Scripting - Control Flow (cont)

```
do
  echo "This is iteration $i"
done
# --- for-loop with a range --- #
for ((i=0; i<=100; i=i+3))
do
  echo $i
done
```

2.4 Shell Scripting - Useful techniques

`$#`	number of arguments input by user
`$0; $1; $2`	1st argument, 2nd argument, etc

`cp file123 fileabc 1> /dev/null 2> &1`

`/dev/null` is system dust bin

`&1` is the standard output

`echo "$0:error:Copy failed" >&2`

`&2` is error output

Shell Script Conditions

String comparison

`[ "$s" ]`	iff length of s is non-zero
`[ "s1" == "$s2" ]`	
`[ "$s1" != "$s2" ]`	

Shell Script Conditions (cont)

`[ "$s1" /> "$s2" ]`	s1 sorted after s2
`[ "$s1" /< "s2" ]`	
File Checking	
`[ -e $file ]`	iff exists
`[ -f $file ]`	iff is a file
`[ -d file ]`	iff is a directory
Number Comparison	
`[ $a -eq $b ]`	iff a = b
`[ $a -ne $b ]`	iff a!=b
`[ $a -lt $b ]`	iff a<b
`[ $a -le $b ]`	iff a<=b
`[ $a -gt $b ]`	iff a>b
`[ $a -ge $b ]`	iff a>=b

Iterate words in a file

```
list=`cat wordlist.txt`
for line in $list
do
  echo "$line"
done
```

C++ Misc

`void func(int array[]);` // array as parameter

C++ Dynamic Array

```
int * a = NULL; int n;
cin >> n; a = new int[n];
...
delete[] a; // free memory
```

Pointers C++

`int *baz`	define a pointer
`&foo`	address of foo
`*baz`	value pointed by baz
`void PBV(int *p)`	parameter is a pointer
`void PBR(int *&p)`	parameter is address of pointer

modifying the parameter modifies the original variable

C++ Linked List and Various functions

```
int main(){ Node *head = NULL; ...
}
void headInsert(Node *&head, int k,
int v){
    Node *newNode = new Node;
    newNode -> key = k;
    newNode -> value = v;
    newNode -> next = head;
    head = newNode;
```

C++ Linked List and Various functions (cont)

```
}
void printList(Node *head){
    Node *current = head;
    while (current!=NULL){
        cout << "Key:" << current-
>key << ",value:" << current-
>value << endl;
        current = current->next;
    }
}
bool isSorted(Node *head){
    Node *current = head;
    Node *previous = NULL;
    while (current!=NULL){
        if (previous !=NULL){
            if (previous->key >
current->key)
                return false;
        }
        previous = current;
        current = current->next;
    }
    return true;
}
void insertInOrder(Node *&head, int
k){
    Node *newNode = new Node;
    newNode->key = k;
```

C++ Linked List and Various functions (cont)

```
    if (head == NULL)
    {
        newNode->next = NULL;
        head = newNode;
    }
    else
    {
        Node *current= head;
        Node *previous = NULL;
        while(current!=NULL)
        {
            if (current->key > k)
                break;
            previous = current;
            current=current->next;
        }
        newNode->next = current;
        if (previous!=NULL)
            previous->next =
newNode;
        else
            head = newNode;
    }
}
```

4. Separate Compilation and Makefile

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

C++ Traverse Linked List using for-loop

```
void TraverseList(Node *head){
  for(Node *n = head; n -> next !=
NULL; n = n -> next){//do
something to n}
}
```

C++ Operator overloading

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

C AVL Tree - Node and maximum

```
struct treeNode {
  int key; struct treeNode * left;
  struct treeNode * right;
}
typedef struct treeNode treeNode;
int maximum(int a, int b){
  return (a > b ? a : b);
}
```

C AVL - Rotation Functions

```
treeNode* R_rotation(treeNode
*parent){
    treeNode *child = parent ->
left;
    parent -> left = child ->
right;
    child -> right = parent; return
child;}
treeNode* L_rotation(treeNode
*parent){
    treeNode *child = parent ->
right;
    parent -> right = child ->
left;
    child -> left = parent; return
child;}
treeNode* LR_rotation(treeNode
*parent){
    treeNode *child = parent ->
left;
    parent -> left =
L_rotation(child);
    return R_rotation(parent);}
treeNode* RL_rotation(treeNode
*parent){
    treeNode *child = parent ->
right;
    parent -> right =
R_rotation(child);
    return L_rotation(parent);}
```

C AVL - Insert()

```
treeNode* Insert(treeNode
*currentNode, int key){
  if(currentNode == NULL){
    currentNode =
(treeNode*)malloc(sizeof(treeNode))
;
    currentNode -> key = key;
    currentNode -> left =
currentNode -> right = NULL;
  }
  else if(key > currentNode ->
key){
    currentNode -> right =
Insert(currentNode -> right, key);
    currentNode =
balance_tree(currentNode);
  }
  else if(key < currentNode ->
key) {
    currentNode -> left =
Insert(currentNode -> left, key);
    currentNode =
balance_tree(currentNode);
  }
  else {
    printf("fail! - duplicated key
\n");
    exit(-1);
  }
  return currentNode;
}
```

C AVL - get_height

```
int get_height(treeNode
*currentNode)
{
  if(currentNode == NULL)
    return 0;
  else{
    int height = 1 +
maximum(get_height(currentNode-
>left), get_height(currentNode-
>height));
    return height;
  }
}
```

C AVL - getBalance()

```
int get_balance(treeNode *
currentNode){
  if(currentNode == NULL) return 0;
  else return
get_height(currentNode->left) -
get_height(currentNode->height);
}
```

C AVL - balance_tree()

```
treeNode* balance_tree(treeNode *
currentNode){
  int height_diff =
get_balance(currentNode);
  if(height_diff > 1)
  {
    if(get_balance(currentNode ->
left) > 0){
      currentNode =
R_rotation(currentNode);
    } else {
```

C AVL - balance_tree() (cont)

```
      currentNode = LR_rotation(currentNode);
    }
  } else if(height_diff < -1){
    if(get_balance(currentNode->right) < 0){
    currentnode = L_rotation(currentNode);
    } else {
      currentNode = RL_rotation(currentNode);
    }
  }
  return currentNode;
}
```

C AVL - main()

```
int main(){
  treeNode *root = NULL; root = Insert(root, 5);}
```

5.1 Containers

`vector<int> v;`	vector definition
`v[i]`	i-th item in the vector
`v.pop_back()`	remove last item
`v.size()`	size of vector
`list<int> l;`	list definition
`l.push_front()`	insert item at front

5.1 Containers (cont)

`l.push_back()`	insert item at back
`l.pop_front()`	remove the first item
`l.pop_back()`	remove last item
`l.front()`	access the first item
`l.back()`	access the last item
`l.size()`	return num of items
`map<K, V> m;`	map definition
`m[i]`	i-th item in the list
`m.count(k)`	return no. of pairs in the map with key = k
`m.size()`	no. of items

Using map with user define objects

```
bool operator<(const Record& a, const Record& b){
  return a.name < b.name;
}
```

Must overload "<" operator

Directives for STL

`#include<vector>; #include<list>; #include<map>; #include<algorithm>;`

Algorithms

Sorting

`sort(v.begin(), v.end())` // vector

`sort(a, a+10);` // array

`c.sort()` // list and maps

`sort(v,begin(), v.end(), compare);`// descend

`bool compare(int a, int b){ return a > b; }`

overload `operator<()` for special tricks

Binary Search

`binary_search(v.begin(), v.end(), target); //returns bool`

"target" is what you are looking for in v

Upper & lower bound

`lower_bound(v.begin(), v.end(), target); // returns ForwardIterator"

`upper_bound(v.begin(), v.end(), target);`

`lower_bound()` returns the earliest postion

`upper_bound()` returns the lastest position

`binary_search()`, `upper_bound()` and `lower_bound()` can be used with `vectors`, `lists`, and `maps`

Random Shuffle (see appendix)

need `<cstdlib>`, `<ctime>`, `srand(time(NULL))`

C++ STL Template Class

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

C++ Template Operator Overloading (One-to-one)

```
template <class T>
class MyCollection{
    vector<T> data;
  public:
    void Add(T const &);
    T & Draw();
  friend ostream &
operator<<(ostream & cout, const
MyCollection<T> &q){
    cout << "Collection" << endl;
    typename
vector<T>::const_iterator itr;
       for(itr =q.data.begin(); itr
!= q.data.end(); itr++)
          cout << " " << *itr <<
endl;
          return cout;
```

C++ Template Operator Overloading (One-to-one) (cont)

```
    }
}
```

C++ Template Overloading (many-to-many)

```
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;
  ... (same)
}
```

C Conversion Specifier

int	%d
float	%f
double	%lf
char	%c
string	%s

C string

`char name[] = "Alan";`

`char name[100]; scanf("%s", name);`

`#include<string.h>`	more functions
`strcopy(char s1[], char s2[])`	copy s2 to s1
`strcat(char s1[], char s2[])`	append s2 to end of s1
`strcmp(char s1[], char s2[])`	return -ive if s1<s2. return +ive if s1>s2. return 0 if s1==s2
`strlen(char s1[])`	return length of string

C Functions

Pass by reference

`void swap(double *a, double *b) {...}`

Using this function: pass the address

`swap(&a, &b);`

C Arrays

```
int array[] = {1, 2, 3};
```

`a[i]` is the same as `*(a+i)`

C Memory Allocation

```
int size; int *a; scanf("%d", &size);
a = malloc(size*sizeof(int));
free(a); // a is pointer
```

C Structure and typedef

```
struct student { char name[20]; int uid; }
typedef struct student student;
int main(){ student a; ... }
```

Python

`int(a); float(a); str(a)`	type casting
`a+b`	concatenation
`s[i]`	access i-th
`s[1:5]`	substring s[1] to s[4]
`s[1:]`	s[1] to end
`s[:4]`	start to s[3]
`len(s)`	length of the string
`print "howdy!",`	print without newline
`s = input("prompt")`	take user input as strings

Python File Input

`with open("filename", "mode") as f:`	open file. define scope
`f.close()`	end of scope
`s = f.read();`	read from file
`f.write(str(a));`	write to the file
file modes	
r	read only
r+	reading and writing
w	write only
w+	writing and reading (overwrite)
a	appending
a+	appending and reading

Python Flow of Control

```
#if-else statement
if condition:
  statement
elif condition:
  statement
else:
  statement
```

Conditions are not enclosed by brackets

Python Logical operators

&&	and
||	or
!	not

Python For-loops

```
for i in list:
  statement
  statement
#Example 1
i = 1
for dir in [ "n", "e", "w", "s" ]:
  print "the" + str(i) + "-th direction is" + dir
  i+=1
#Example2
for i in range(0, 71):
  if i % 7 == 0:
    print i,
```

Python Array

`arr = [0] * i`	empty array with size i
`arr[i]`	i-th item in array