Set up

makefile

```
maze: main.o maze.o
	g++ -Wall -g main.o maze.o -o
maze

%.o: %.c
	g++ -Wall -g -c $<

main.o: maze.h

maze.o: maze.h

clean:
	rm -f *.o maze
```

header.h

```
#ifndef MAZE_H
#define MAZE_H

bool isVowel(char ch);

#endif
```

file.cpp

```
#include <iostream>
#include <cstring>
#include <cstdlib>
#include <cctype>
.
.
.
#include "header.h"
```

Casting

`int to char`	`char x = (char)(a+'0'); static_cast<char>(a);`
`int to string (#include <string>)`	`string str = to_string(number);`
`cstring to int (#include <cstdlib>)`	`int num = atoi(const char* str)`

Strings

Library	**`<cstring>`**	**`<string>`**
Initialisation	`char str[] = "xyz";` `char str[size]; str[0]='\0';`	`string str = "xyz";` `string str(str1);`
Assignment	`strcpy(str, str1);` `strncpy(str, str1, n); // n characters`	`str = str1;`
Concat	`strcat(str,str1);`	`+`
Access	`str[index];`	`str.at(i);`
Adding/removing string	`n = strlen(str);` `str[n] = 'A'; str[n+1] = '\0';` `strncpy(buf, string + pos, len);` `// copy substring from pos to len`	`str.push_back(letter);   //char letter` `str.insert(pos,str1);   // adds a string` `str.erase(pos,length);   //removes`
Comparison	`strcmp(str,str1); //returns 0 if true` `strncmp(str,str1,n); // compares first n chars`	`str == str1;`
Length	`strlen(str);       //` **`excl '\0'`** `sizeof(str);       // incl '\0'`	`str.length();  //excl '\0'`
Using Tokens	`char *sptr = strtok(string1, ", .!");` `while(sptr != NULL) {sptr = strtok(NULL, ", .!");`	`//`
Print	`cout << str;`	`cout << str;`
I/O	`(c)in.getline(cstring,MAX);`	`getline((c)in, line);` `((c)in.ignore(n,'\n');`
Returning string from a function	`char *astr = new char[512];` `return astr;`	

I/O

Library(ies)	**`<iostream>`**	**`<fstream> && <cstdlib>`**
Initialisation	`istream in;  //declare with std::` `ostream out;`	`ifstream in("input.txt"); in.fail();` `in.close();` `ofstream out("output.txt"); out.close();`
Formatted	`(c)in >> / (c)out << // skips whitespaces, starts new input at ws or '\n'`	
Unformatted	`in.get(ch); || out.put(ch);` `in.get(charArray, size); in.getline(charArray, size)` `(c)out.put(letter);`	
other	`(c)in.putback(ch);    // return character to stream`	

- loops
  - for individual chars with whitespace: `char letter; while (in.get(letter)) {;}`
  - for words /skipping whitespace: `char word[]; while (in >> word) {;}`

Pointers

	Pointer	Dynamic array	Array of pointers
Initialisation	`int *p = new int; // NULL` `*p = &num;`	`int *p = new int[];`	`int *p[size] = {"ab" , "cd"}; int *p[size];`
Assignment	`*p = number;`	`for (…) p[n]=num;`	`for (…) p[n]=sth;`
Destroy	`delete p;`	`delete [] p; // whole thing`	`-    //not dynamic mem!`

static variables in functions persist between calls (lives on the heap).

`*&` is a reference to a pointer (referring to a pointer type)

Linked list

Node definition

```
struct Node {
            string content;
            Node* ptr;
};
typedef Node* NodePtr;
```

Add node

```
current_node->ptr= head;
head = current_node;
```

Recursion

- Use bool type function if possible (helper functions) if stuck >> if, else
- Consider mechanism to change function parameter(s):
  - Pointer to next char in string i.e. compare(&str1[1], &str2[1]);
- Return value of recursive function as part of the function

Short-hand if-else:

- `x <3 ? (true branch) : (false branch);`

ASCII Table (Important ones only)

Dec	Out
32	‘ ’
48	0
57	9
65	A
90	Z
97	a
122	z

Extras

Selection Sort

```
// function to implement selection sort on a string
void selSort(char* str) {
        char temp;
        int i, j, n = strlen(str);
        int currentMin;

        // convert all to upper case
        for (i = 0; i < n; i++) {
                str[i] = toupper(str[i]);
        }

        // selection sort algorithmm
        for (i = 0; i < n - 1; i++) {
                currentMin = i;

                for (j = i + 1; j < n; j++) {
                        if (str[j] < str[currentMin])
                                currentMin = j;
                }

                // swap places
                if (str[currentMin] != str[i]) {
                        temp = str[i];
                        str[i] = str[currentMin];
                        str[currentMin] = temp;
                }
        }
}
```

Converting between strings to cstrings

```
// strings and c-strings
#include <iostream>
#include <cstring>
#include <string>

int main ()
{
  std::string str ("Please split this sentence into
tokens");

  char * cstr = new char [str.length()+1];
  std::strcpy (cstr, str.c_str());

  // cstr now contains a c-string copy of str

  char * p = std::strtok (cstr," ");
  while (p!=0)
  {
    std::cout << p << '\n';
    p = std::strtok(NULL," ");
  }

  delete[] cstr;
  return 0;
}

// convert cstring to string
string str1 = cstr;
```