Intro. to Computer Programming
Midterm 2

CHEN Xiwen

UM-SJTU Joint Institute

July 16, 2019

Table of contents

Open/Close Files

Syntax.

```
1 FILE* fp = fopen(filename, mode);
2 fclose(fp);
```

Mode	Explanation
`r`	read from **existing** file
`w`	open or **create** a file to write
`a`	open or **create** a file, append data
`r+`	read from and write to an **existing** file
`w+`	open or **create** a file to read and write, discarding existing contents
`a+`	open or **create** a file to read and write, append to existing contents

Figure: File I/O Modes

Read from / Write to a File

Syntax.

```
1 int fscanf(FILE* restrict stream, const char* format,
      ...);
2 int fprintf(FILE* restrict stream, const char* restrict
      format, ...);
```

Example. Read integers from a file, space separated, until \n is met.

```
1 int num;
2 FILE* fp = fopen("input.txt", "r");
3 while (fscanf(fp, "%d ", &num) != EOF) {
4     printf("%d\n", num);
5 }
```

String Operations

Useful Functions.

```
size_t strlen(const char* s);
char* strcpy(char* dst, const char* src);
char* strcat(char* restrict s1, const char* restrict s2);
int strcmp(const char* s1, const char* s2);
```

Example. Command line arguments.

```
if (strcmp(argv[i], "-h") == 0) {
    // do something;
}
```

Dynamic Memory

Syntax.

```
1 void* malloc(size_t size);
2 void* calloc(size_t count, size_t size);
3 void* realloc(void* p, size_t size);
4 void free(void* ptr);
```

Common Mistakes

Mistake 1. Allocate memory in subfunctions, which does not return the pointer nor free the memory.

```
1  int sum(int a, int b) {
2      int* c = malloc(sizeof(int));
3      return a + b;
4  }
5  int main() {
6      int c = sum(2, 3);
7      printf("2 + 3 = %d\n", c);
8      return 0;
9  }
```

Common Mistakes

Mistake 2. Pass a pointer to a subfunction and attempt to allocate memory to the pointer.

```
1  void create_node(node_t* node) {
2      node = malloc(sizeof(node_t));
3  }
4  int main() {
5      node_t* node;
6      create_node(node);
7      // do something;
8      return;
9  }
```

Correction. Pass a pointer to pointer.

```
1  void create_node(node_t** node) {
2      *node = malloc(sizeof(node_t));
3  }
```

Common Mistakes

Mistake 3. Dynamic arrays in structures.

```
1  typedef struct Player {
2      card_t* cards;
3  } player_t;
4  int main() {
5      player_t* players = malloc(3 * sizeof(player_t));
6      for (int i = 0; i < 3; i++) {
7          players[i].cards = malloc(52*sizeof(card_t));
8          // do something;
9      }
10     // do something;
11     free(players);
12     return 0;
13 }
```

Common Mistakes

Mistake 4. Double free of memory.

```
1  int main() {
2      int* a = malloc(10 * sizeof(int));
3      int* b = realloc(a, 7 * sizeof(int));
4      // do something;
5      free(a);
6      free(b);
7  }
```

Common Mistakes

Mistake 5. Lose the address of allocated memory.

```
1  int main() {
2      int* a = malloc(10 * sizeof(int));
3      int b[5] = {1, 2, 3, 4, 5};
4      a = b;
5      // do something;
6      free(a);
7  }
```

Common Mistakes

Mistake 6. Misunderstand "pointer" and "memory" pointed by the pointer.

```
1  void clear_list(node_t* head) {
2      node_t* tmp_node = malloc(sizeof(node_t));
3      tmp_node = head;
4      // do something;
5  }
6  int main() {
7      node_t* head = create_list();
8      // do something;
9      clear_list(head);
10     return 0;
11 }
```

Clarifications

- The number we use `malloc` or `calloc` should match the number of `free` in the program.
- To change the address that is stored in the pointer, pass a pointer to pointer to the subfunction. Otherwise, we are actually passing a *copy* of the address to the subfunction.
- In case of array of pointers, where each pointer points to a dynamic memory, we need to traverse the array to free each dynamic memory.

Remarks

1. Start from the easiest question.
2. Do not waste too much time on a single question, especially if it does not worth too many marks.
3. Use templates.
4. If you have a clear solution to the problem, do not spend too much time on README.
5. `C` is supposed to be much easier than `MATLAB` for you. Therefore, be confident!
6. Practice yourself to achieve 5.

Good luck for your Midterm 2!