Part 1

P1Q1:

Uniprogramming:

1. Only one program is present in the memory at a time, thus only one program is executed at a time
2. Resources are provided to support this single program which is running, thus the resources do not need to be dynamically allocated
3. The usage of the memory at certain time is more likely less than Multiprogramming for it is one program loaded at a time

Multiprogramming:

1. More than one programs can be present in the memory, and although one program is executed at a time, the wait time is significantly less than Uniprogramming since the execution of a second program may happen when the system is waiting for the resource of the first program
2. The resources which are used by the programs are required to be dynamically allocated to prevent resources usage conflicts and mistaken modification
3. Since more than one programs are often loaded at certain time, the memory usage at that given time are more likely bigger than Uniprogramming

Time-Sharing System:

1. Multiple users with different terminals may use the same system, and multiple jobs are executed by CPU by switching between them
2. The same CPU may be shared among multiple users, thus a reliable process scheduling for different users and dynamic reource allocation are required
3. Since more than one programs are often loaded at certain time, the memory usage at that given time are more likely bigger than Uniprogramming, and the response time is quick for all the users. However, the reliability is questionable

P1Q2:

Uniprogramming ABC:

(2+10+4)*3 = **48 msec**

Multiprogramming ABC(assume hardware is good enough):

Since 3 programs only need 6 msec for the first run, thus the total time is

10+4*3= **22 msec**

P1Q3:
The modified program will be:

```
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdlib.h>

int main(){
	FILE *fp;
	if ((fp = freopen("redirect.txt", "w+", stdout)) == NULL){
		exit(-1);
	}

	printf("A simple program output\n");
	fclose(fp);
	return 0;
}
```