# C Programming

Noureddine Hamid - Nouredine Ouhaddou - Redone Mahjoubi ${\rm TP}\ 01$ 

## Exercice 1: (Quick Recap)

- 1. In how many bits an integer is stored?
- 2. In how many bytes an integer is stored?
- 3. What would be printed if you run the following program:

```
#include <stdio.h>
int main()
{
    printf("%d\n", 1 >> 1);
    printf("%d\n", 1 << 1);
    printf("%d\n", 1 << 2);
    return (0);
}</pre>
```

- Explain the behavior.
- Explain why shifting an integer by 1, is equivalent to multiplying it by 2.
- 4. What is the range of values, that could be stored in an unsigned char.
- 5. what is the range of values, that could be stored in a char.
- 6. which bit is reserved to represent the sign, in the case of an integers.

### Exercice 2:

Consider the following program:

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

int **dummy(int **x, int n)
{
    int **y;
```

```
int i;
          \mathbf{int} \quad \mathtt{j} \ ;
          i = 0;
          y = (int**) malloc(sizeof(int*) * (n + 1));
          while (i < n)
                     y[i] = (int*) malloc(sizeof(int) * n);
                     j = 0;
                     \mathbf{while}\,(\,\mathrm{j}\ <\ \mathrm{n}\,)
                     {
                               y[i][j] = j;
                               j++;
                     }
                     i++;
          y[n] = x[0];
          return(y);
}
int main()
          int n;
          int **x;
          int i;
          int j;
          n = 10;
          i\ =\ 0\,;
          x = (int**) malloc(sizeof(int*) * n);
          while (i < n)
          {
                     x[i] = (int*) malloc(sizeof(int) * n);
                     j = 0;
                     \mathbf{while}(j < n)
                     {
                               x\,[\;i\;]\,[\;j\;]\;=\;j\;;
                     }
                     i++;
          x = dummy(x, n);
          i = 0;
          \mathbf{while} \ (\, i \ < \, n \ + \ 1\,)
                     j = 0;
                     while (j < n)
                                printf("\%d\_"\;,\;\;x[\;i\;][\;j\;]);
                               j++;
                     }
                     printf("\n");
                     i++;
          }
}
```

Using valgrind, hunt down every memory leak and eliminate it.

### Exercice 3:

- 1. Put your .c files and .h files in the directories srcs and includes respectively.
- 2. Create a makefile that compiles bitwise functions into a static library.

The makefile should contain the following:

- A variable which holds the names of your .c files.

  Note: for a multi line variable definition use backslash at the end of the line.
- A variable which holds the name of your library.
- A default rule 'all', which executes when we run make without specifying any rule name.
- A rule with the same name of your library, that compiles your library.
- A rule 'clean', that removes .o files.
- A rule 'fclean' (full clean), that removes .o files and the library Note: fclean depends on clean, so consider making it a dependency
- A rule re, that recompiles your library.

### Exercice 4:

Write a program to store n elements in an array, then print the elements using pointer arithmetic only (don't use array indexing).