PARCIAL N° 1

1. **(1 punto)** Indicar la salida del siguiente programa, si hay errores indicarlos.

```
char bad_hope[]="Washing your car to make it rain doesn't work.";
void print_token(char *p);

int main(void)
{
    char *p=bad_hope;

    print_token(p+8);
    print_token(&p[13]);
    print_token(bad_hope+33);
    print_token(&bad_hope[41]);

    return 0;
}

void print_token(char *p)
{
    while(*p!=' '&& (*p))
        putchar(*p++);
    putchar(' ');

}
```

_____

_____

_____

_____

2. (2 puntos) Implementar una función que imprima en binario un número de 16 bits. El prototipo de la función es **void print_binary(uint16_t number)**. El formato de impresión de la salida debe seguir el ejemplo que se muestra más abajo. Se debe imprimir un espacio cada 4 bits. **No** se deberá usar **printf()** en su lugar usar **putchar()**. Se prestara especial atención a la estructura del código.

```
int main(void)   //Test Bench
{
        uint16_t number=0b01110001;    //Test Number 1
        print_binary(number);

        number=0xABCD;                 //Test Number 2
        print_binary(number);

        number=100;                    //Test Number 3
}       print_binary(number);
```

Ejemplo de salida:

Number in binary is: 0000 0000 0111 0001
Number in binary is: 1010 1011 1100 1101
Number in binary is: 0000 0000 0110 0100

3. (2 puntos) Implementar una función que imprima un número en hexadecimal. El prototipo de la función es:

void print_hex_word(uint16_t word)

No se deberá usar **printf()** en su lugar usar **putchar()**. **Se prestara especial atención a la modularizacion del código (la función NO debe hacer todo).**

Ejemplo: print_hex_word(0xAB23); → AB23

4. (1 punto) Cuál es la salida del siguiente programa?.Justificar la respuesta

```
unsigned char x=255;
unsigned char y=1;

int suma ( char x,char y);

int main(void)
{
    printf("Suma:%d \n",suma(x,y));    RESPUESTA:
    return 0;
}

int suma ( char x,char y)
{
      return(x+y);
}
```

5. (1 punto) Cuál es la salida del siguiente programa?.Justificar la respuesta

```
int M2D[][5]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};
int *p0=M2D[0];
int *p1=M2D[1];

int main(void)
{
    printf("S1: %d \n",p0[4]);  RESPUESTA:
    printf("S2: %d \n",p1[5]);  RESPUESTA:
    return 0;
}
```

6. **(2 puntos)** Dado el siguiente arreglo:

char *nombres[] = {"Alan", "Frank","Mary"}; /* Declarado en main */

Se pide escribir una función **swap**(....) capaz de intercambiar un par de elementos del arreglo "nombres". A continuación escribir un main que intercambie el 1er y 3er elemento del mismo. Se deberá imprimir todo el arreglo antes y después de llamar a **swap(....)**. El código que imprime debe ser independiente del largo del arreglo.

7. **(1 punto)** Indicar la salida del siguiente programa:

```
#include <stdio.h>
size_t strlen(const char *str);

int main()
{                                        RESPUESTA:
   char *s="EMIT";
   char r[100];
   int n, c, d;

   n = strlen(s);

   for (c = n - 1, d = 0; c >= 0; c--, d++)
      r[d] = s[c];

   r[d] = '\0';

   printf("%s\n", r);

   return 0;
}
size_t strlen(const char *str)
{
      size_t n=0;

            while (*str++)
                  n++;
            return(n);
}
```