Cada uno de los trozos de programas tienen, al menos un error y/o alguna construcción cuestionable o no portable.
Identifíquelas y de alternativas para su re-escritura.

**(i)**

```
typedef int NUMBER;
typedef int COUNT;
COUNT index (NUMBER a[], COUNT n, NUMBER m) {
     COUNT i;
     for (i=0; i<n; i++)
     if (m == a[i]) return m;
     return n;
}
```

**(ii)**

```
int main() {
     int *ptr, a=34;
     int *ptr2=&b, b=45;
     *ptr = 34;
     printf("%d =?= %d", a, *ptr);
     return 0;
}
```

**(iii)**

```
#include <string.h>
int main () {
     int *p = malloc(20*sizeof(int))
     memset(p, 20, 'a');
     free(p);
     return 0;
}
```

**(iv)**

```
int main() {
     int *pi = malloc(sizeof(int));
     int *pj = malloc(sizeof(int));
     *pi = 11;
     pj = pi;
     printf("*pi=%d, *pj=%d", *pi, *pj);
     free(pj);
     // resto de código ...
     return 0;
}
```

**(v)**

```
#include <stdio.h>
int a, b;
void initialize() {
     if (a>0) b=1; else b=0;
}

int main() {
     a=1;
     initialize;
     printf("%d %d\n", a, b);
     return 0;
}
```

**(vi)** Señalar de que manera se puede hacer más eficiente la siguiente función

```
int esPrimo(unsigned a) {
     /* precondición: a > 0 */
     /* retorna 1 si a es primo, 0 en caso contrario */
     int i, flag = 1;
     for (i = 2; i < a; i++)
          if (!(a % i)) flag = 0;
     }
     return flag;
}
```

**(vii)**

```
struct vector {
     float x, y;
}

struct vector *normal(struct vector v) {
     struct vector *ptr;
     ptr->x = v.x;
     ptr->y = v.y;
     return ptr;
}
```

**(viii)**

```
#define MAX(A,B) ((A)>(B)?(A):(B))
int main() {
     int i=8, j=4;
}
printf("Que entero es mayor? %d o %d?", i, j);
printf("\nEl mayor es %d", MAX(i++, j++));
```