Partiel QCM - Langage C (LC4)

1er avril 2009

Durée: 1h45 - Documents interdits

Pour chaque question, il y a exactement une bonne réponse. Le tableau où vous devez répondre est fourni séparément.
+1 pour une bonne réponse, 0 pour absence de réponse, -0,25 pour une mauvaise réponse.
Vous devez obligatoirement répondre sur le tableau fourni. Pensez à le rendre avant de partir ;-)

Question 0. Soit `ex0.c` le code suivant :

```
#include <stdio.h>

int main() {
  int a = 1;
  int b = 2;
  printf("a = %d, b = %d\n", b, a);
  return 0;
}
```

On compile avec `gcc ex0.c` et si un exécutable est créé on l'exécute. Le programme `ex0.c` :

1. ne compile pas
2. provoque une erreur à l'exécution (erreur de segmentation par exemple)
3. affiche `a = 1, b = 2`
4. affiche `a = 2, b = 1`

Question 1. Soit `ex1.c` le code suivant :

```
int main() {
  int i = '0';
  printf("%d\n", i);
  return 0;
}
```

On compile avec `gcc ex1.c` et si un exécutable est créé on l'exécute. Le programme `ex1.c` :

1. ne compile pas
2. provoque une erreur à l'exécution (erreur de segmentation par exemple)
3. affiche 0 à l'exécution
4. affiche 48 à l'exécution

Question 2. Soit `ex2.c` le code suivant :

```c
#include <stdio.h>

int main() {
  int i = 0;
  printf("%d\n", i++);
  return 0;
}
```

On compile avec `gcc ex2.c` et si un exécutable est créé on l'exécute. Le programme `ex2.c` :

1. ne compile pas
2. provoque une erreur à l'exécution (erreur de segmentation par exemple)
3. affiche 0 à l'exécution
4. affiche 1 à l'exécution
5. affiche 2 à l'exécution

Question 3. Soit `ex3.c` le code suivant :

```c
#include <stdio.h>

int main() {
  int i = 0;
  int *ptr;
  *ptr = i;
  printf("%d \n", ++*ptr);
  return 0;
}
```

On compile avec `gcc ex3.c` et si un exécutable est créé on l'exécute. Le programme `ex3.c` :

1. ne compile pas
2. provoque une erreur à l'exécution (erreur de segmentation par exemple)
3. affiche 0 à l'exécution
4. affiche 1 à l'exécution

Question 4. Soit `ex4.c` le code suivant :

```c
#include <stdio.h>

int main() {
  int i;
  int *ptr = &i;
  printf("%x\n", &*ptr);
  return 0;
}
```

On compile avec `gcc ex4.c` et si un exécutable est créé on l'exécute. Le programme `ex4.c` :

1. ne compile pas
2. provoque une erreur à l'exécution (erreur de segmentation par exemple)

3. affiche l'adresse de `i` en hexadécimal
4. affiche l'adresse de `ptr` en hexadécimal

Question 5. Soit `ex5.c` le code suivant :

```
#include <stdio.h>

int main() {
  int i, j = 0;
  i = j = 6;
  printf("i = %d, j = %d\n", i, j);
  return 0;
}
```

On compile avec `gcc ex5.c` et si un exécutable est créé on l'exécute. Le programme `ex5.c` :

1. ne compile pas
2. provoque une erreur à l'exécution (erreur de segmentation par exemple)
3. affiche `i = 6,j = 6`
4. affiche `i = 106,j = 6`
5. affiche `i = 0,j = 6`

Question 6. Soit `ex6.c` le code suivant :

```
#include <stdio.h>

#ifdef X

int main() {
  printf("def\n");
  return 0;
}

#else

int main() {
  printf("ndef\n");
  return 0;
}

#endif
```

On compile avec `gcc ex6.c` et si un exécutable est créé on l'exécute. Le programme `ex6.c` :

1. ne compile pas
2. provoque une erreur à l'exécution (erreur de segmentation par exemple)
3. affiche `def`
4. affiche `ndef`

Question 7. Soit `ex7.c` le code suivant :

```
#include <stdio.h>

#ifdef X

int main() {
  printf("def\n");
  return 0;
}

#else

int main() {
  printf("ndef\n");
  return 0;
}

#endif
```

On compile `ex7.c` avec `gcc -D"X" ex7.c` et si un exécutable est créé on l'exécute. Le programme `ex7.c` :

1. ne compile pas
2. provoque une erreur à l'exécution (erreur de segmentation par exemple)
3. affiche `def`
4. affiche `ndef`

Question 8. Soit `ex8.c` le code suivant :

```
#include <stdio.h>

#ifdef X

int main() {
  printf("def, %d\n", X);
  return 0;
}

#else

int main() {
  printf("ndef\n");
  return 0;
}

#endif
```

On compile avec `gcc -DX="1" ex8.c` et si un exécutable est créé on l'exécute. Le programme `ex8.c` :

1. ne compile pas

2. provoque une erreur à l'exécution (erreur de segmentation par exemple)
3. affiche `def, 1`
4. affiche `ndef`

Question 9. Soit `ex9.c` le code suivant :

```
#include <stdio.h>

int main() {
  int a, b;
  int *ptr1, *ptr2;
  a = 5;
  b = a;
  ptr1 = &a;
  ptr2 = ptr1;
  b = (*ptr2)++;
  printf("a = %d, b = %d, *ptr1 = %d, *ptr2 = %d\n", a, b, *ptr1, *ptr2);
  return 0;
}
```

On compile avec `gcc ex9.c` et si un exécutable est créé on l'exécute. Le programme `ex9.c` :

1. ne compile pas
2. provoque une erreur à l'exécution (erreur de segmentation par exemple)
3. affiche `a = 6, b = 5, *ptr1 = 6, *ptr2 = 6`
4. affiche `a = 5, b = 5, *ptr1 = 5, *ptr2 = 5`
5. affiche `a = 6, b = 6, *ptr1 = 6, *ptr2 = 6`

Question 10. Soit `ex10.c` le code suivant :

```
#include <stdio.h>

int main() {
  int a, b;
  int *ptr1, *ptr2;
  a = 5;
  b = a;
  ptr1 = &a;
  a++;
  ptr2 = &b;
  b += a = b;
  printf("a = %d, b = %d, *ptr1 = %d, *ptr2 = %d\n", a, b, *ptr1, *ptr2);
  return 0;
}
```

On compile avec `gcc ex10.c` et si un exécutable est créé on l'exécute. Le programme `ex10.c` :

1. ne compile pas
2. provoque une erreur à l'exécution (erreur de segmentation par exemple)
3. affiche `a = 5, b = 10, *ptr1 = 5, *ptr2 = 10`
4. affiche `a = 5, b = 11, *ptr1 = 5, *ptr2 = 11`
5. affiche `a = 6, b = 6, *ptr1 = 6, *ptr2 = 11`

Question 11. Soit `ex11.c` le code suivant :

```c
#include <stdio.h>

void trio(int a, int b, int c) {
  a = b + c;
  b = c + a;
  c = a + b;
}

int main(void) {
  int a = 1, b = 2, c = 3;
  trio(a, b, c);
  printf("%d %d %d\n", a, b, c);
  return 0;
}
```

On compile avec `gcc ex11.c` et si un exécutable est créé on l'exécute. Le programme `ex11.c` :

1. ne compile pas
2. provoque une erreur à l'exécution (erreur de segmentation par exemple)
3. affiche `5 4 3`
4. affiche `5 8 13`
5. affiche `1 2 3`
6. affiche `3 6 9`
7. affiche `1 3 6`

Question 12. Soit `ex12.c` le code suivant :

```c
#include <stdio.h>

void trio(int *a, int *b, int *c) {
  *a = *b + *c;
  *b = *c + *a;
  *c = *a + *b;
}

int main(void) {
    int a = 1, b = 2, c = 3;
    trio(&a, &b, &c);
    printf("%d %d %d\n", a, b, c);
    return 0;
}
```

On compile avec `gcc ex12.c` et si un exécutable est créé on l'exécute. Le programme `ex12.c` :

1. ne compile pas
2. provoque une erreur à l'exécution (erreur de segmentation par exemple)
3. affiche `5 4 3`
4. affiche `5 8 13`
5. affiche `1 2 3`

6. affiche `3 6 9`
7. affiche `1 3 6`

Question 13. Soit `ex13.c` le code suivant :

```
#include <stdio.h>

#define TAB_LENGTH 3

int main() {
  int tab[TAB_LENGTH];
  int j;
  for (j = 0; j < TAB_LENGTH; j++)
    tab[j] = 5;
  j = 0;
  printf("[");
  while (j < TAB_LENGTH)
    printf(" %d ", tab[j]); j++;
  printf("]");
  return 0;
}
```

On compile avec `gcc ex13.c` et si un exécutable est créé on l'exécute. Le programme `ex13.c` :

1. ne compile pas
2. provoque une erreur fatale à l'exécution (erreur de segmentation par exemple)
3. boucle
4. affiche `[ 5 5 5 ]`
5. affiche autre chose

Question 14. Soit `ex14.c` le code suivant :

```
#include <stdio.h>

#define TAB_LENGTH 3

int main() {
  int tab[TAB_LENGTH];
  int j = 0;
  for(; j <= TAB_LENGTH; j++)
    tab[j] = 5;
  printf("[");
  for(j = 0; j <= TAB_LENGTH; j++)
      printf(" %d ", tab[j]);
  printf("]");
  return 0;
}
```

On compile avec `gcc ex14.c` et si un exécutable est créé on l'exécute. Le programme `ex14.c` :

1. ne compile pas
2. fonctionne normalement et affiche `[ 5 5 5 ]`
3. pose des soucis à l'exécution

Question 15. Soit `ex15.c` le code suivant :

```
#include <stdio.h>

#define TAB_LENGTH 3

int main() {
  int tab[TAB_LENGTH];
  int j = 0;
  int *ptr = tab;
  for(; j < TAB_LENGTH; j++)
    tab[j] = 5;
  *(ptr + 1) = 3;
  printf("[ %d %d %d ]\n", tab[0], tab[1], tab[2]);
  return 0;
}
```

On compile avec `gcc ex15.c` et si un exécutable est créé on l'exécute. Le programme `ex15.c` :

1. ne compile pas
2. provoque une erreur fatale à l'exécution (erreur de segmentation par exemple)
3. boucle
4. affiche `[ 5 5 5 ]`
5. affiche `[ 3 5 5 ]`
6. affiche `[ 5 3 5 ]`

Question 16. Soit `ex16.c` le code suivant :

```
#include <stdio.h>

#define TAB_LENGTH 3

int main() {
  int tab[TAB_LENGTH];
  int j = 0;
  int *ptr = &tab[0];
  for(; j < TAB_LENGTH; j++)
    tab[j] = 5;
  *(ptr + 1) = 3;
  printf("[ %d %d %d ]", tab[0], tab[1], tab[2]);
  return 0;
}
```

On compile avec `gcc ex16.c` et si un exécutable est créé on l'exécute. Le programme `ex16.c` :

1. ne compile pas
2. provoque une erreur fatale à l'exécution (erreur de segmentation par exemple)
3. boucle
4. affiche `[ 5 5 5 ]`
5. affiche `[ 3 5 5 ]`
6. affiche `[ 5 3 5 ]`

Question 17. Soit `ex17.c` le code suivant :

```c
#include <stdio.h>

#define TAB_LENGTH 3

int main() {
  int tab[TAB_LENGTH];
  int j = 0;
  int *ptr = &tab[3];
  for(; j < TAB_LENGTH; j++)
    tab[j] = 5;
  *(tab + 1) = 3;
  *(ptr - 1) = 3;
  printf("[ %d %d %d ]", tab[0], tab[1], tab[2]);
  return 0;
}
```

On compile avec `gcc ex17.c` et si un exécutable est créé on l'exécute. Le programme `ex17.c` :

1. ne compile pas
2. provoque une erreur fatale à l'exécution (erreur de segmentation par exemple)
3. boucle
4. affiche `[ 3 5 3 ]`
5. affiche `[ 3 3 5 ]`
6. affiche `[ 5 3 3 ]`
7. affiche `[ 5 3 5 ]`

Question 18. Soit `ex18.c` le code suivant :

```c
#include <stdio.h>

struct list_v {
  struct list_v *suivant;
};

typedef struct list_v *list;

int main() {
  list m;
  m->suivant = NULL;
  return 0;
}
```

On compile avec `gcc ex18.c` et si un exécutable est créé on l'exécute. Le programme `ex18.c` :

1. ne compile pas
2. provoque une erreur fatale à l'exécution (erreur de segmentation par exemple)
3. boucle
4. fonctionne normalement mais n'affiche rien

Question 19. Soit `ex19.c` le code suivant :

```
#include <stdio.h>

struct list_v {
  int i;
  struct list_v *suivant;
};

int main() {
  struct list_v maliste;
  maliste.suivant = maliste;
  return 0;
}
```

On compile avec `gcc ex19.c` et si un exécutable est créé on l'exécute. Le programme `ex19.c` :

1. ne compile pas
2. provoque une erreur fatale à l'exécution (erreur de segmentation par exemple)
3. boucle
4. fonctionne normalement mais n'affiche rien

Question 20. Soit `ex20.c` le code suivant :

```
#include <stdlib.h>

struct list_couple {
  int membre1;
  int membre2;
  struct list_couple *suivant;
};

typedef struct list_couple *list;

int main() {
  list m = malloc(sizeof(struct list_couple));
  list m2 = malloc(sizeof(struct list_couple));
  m->suivant = m2;
  free(m);
  return 0;
}
```

On compile avec `gcc ex20.c` et si un exécutable est créé on l'exécute. Le programme `ex20.c` :

1. ne compile pas
2. provoque une erreur à l'exécution (erreur de segmentation par exemple)
3. fonctionne et avant le retour de la fonction `main()` tout l'espace occupé par `m` et `m2` a été libéré.
4. fonctionne mais avant le retour de la fonction `main()` tout l'espace occupé par `m` et `m2` n'a pas été libéré.

Question 21. Soit `ex21.c` le code suivant :

```
#include <stdlib.h>

struct list_couple {
  int membre1;
  int membre2;
  struct list_couple *suivant;
};

typedef struct list_couple *list;

int main() {
  list m = malloc(sizeof(struct list_couple));
  list m2 = malloc(sizeof(struct list_couple));
  (*m).suivant = *m2;
  free(m);
  return 0;
}
```

On compile avec `gcc ex21.c` et si un exécutable est créé on l'exécute. Le programme `ex21.c` :

1. ne compile pas
2. provoque une erreur à l'exécution (erreur de segmentation par exemple)
3. fonctionne et avant le retour de la fonction `main()` tout l'espace occupé par `m` et `m2` a été libéré.
4. fonctionne mais avant le retour de la fonction `main()` tout l'espace occupé par `m` et `m2` n'a pas été libéré.

Question 22. Soit `ex22.c` le code suivant :

```
#include <stdlib.h>

struct list_couple {
  double membre1;
  double membre2;
  struct list_couple *suivant;
};

typedef struct list_couple *list;

int main() {
  list m = malloc(sizeof(struct list_couple));
  list m2 = malloc(sizeof(struct list_couple));
  m->membre1 = 18.5;
  m->membre2 = 23.4;
  m->suivant = m2;
  free(m->membre1);
  free(m->membre2);
  free(m2);
  free(m);
  return 0;
}
```

On compile avec `gcc ex22.c` et si un exécutable est créé on l'exécute. Le programme `ex22.c` :

1. ne compile pas
2. fonctionne et avant le retour de la fonction `main()` tout l'espace occupé par `m` et `m2` a été libéré
3. fonctionne mais avant le retour de la fonction `main()` tout l'espace occupé par `m` et `m2` n'a pas été libéré.

Nom :

Prénom :

Groupe :

Pour chaque question q0, q1, q2 etc. cochez par une croix les cases correspondant aux réponses que vous estimez justes. Votre réponse doit être lisible et sans ambiguïté. N'écrivez pas en rouge, cette couleur est réservée aux correcteurs.

	r1	r2	r3	r4	r5	r6	r7	score
q0								
q1								
q2								
q3								
q4								
q5								
q6								
q7								
q8								
q9								
q10								
q11								
q12								
q13								
q14								
q15								
q16								
q17								
q18								
q19								
q20								
q21								
q22								
							total	