Prova scritta Programmazione I - 12 Novembre 2018.

Nome e Cognome: _____ Matricola: _____

1. 3 punti Elencare le conversioni di tipo, specificando se implicite o esplicite. Dato *USHRT_MAX = 65535*, scrivere il valori di *d* prima dell'uscita da *main*.

```
1 int fun(long int a) {
2   return (a * 'a' * 3); }
3
4 int main(void) {
5   unsigned a =  -1;
6   float b= fun(a);
7   float *p= (float*) malloc(sizeof(
      float));
8   unsigned short d = -10;
9 }
```

Conversioni implicite
Linea 5: -1 da int a unsigned int
Linea 6: a da uint a long int, il valore di ritorno da int a float
Linea 2: 'a' da int a long int, 3 da int a long int, risultato da int a long int
Linea 8: -10 da int a unsigned short

Conversioni esplicite:
Linea 7: da void* a float*

Valore finale di d= 65526 (USHRT_MAX +1 + -10)

2. 3 punti Scrivere una funzione *main()* che stampi i numeri primi come nella figura sottostante: n è il numero di righe letto da tastiera (nell'esempio, n uguale a 5).

```
2
3 5
7 11 13
17 19 23 29
31 37 41 43 47
```

3. 3 punti Scrivere cosa stampa il seguente programma.

```
1 int a= 0xb;
2 while(a > 15 ? a++: (a--, --a)) {
3   printf("%d \n", a);
4   if (a + 2 >= 0x10)
5     a= -0x4;
6     a+= 3;
7 }
8 !a && a++;
9 printf("a: %d\n", a);
```

9
10
11
12
13
17
-3
-2
-1
a: 1

4. 3 punti La funzione [redacted] *mirror* crea un nuovo array e lo ritorna alla funzione *main* invertendo l'ordine degli elementi passati. Scrivere la funzione *mirror* e finire di scrivere la funzione *main* in esempio.

```
1 int main() {
2   int a[]= {1,4,5,2,1,3};
3   /* Definire il ritorno */ = mirror(
      a, 6);
4   /* Stampare l'array a invertito:
      3,1,2,5,4,1 */
5 }
6
```

```
int* mirror (int*a1, int n) {
 int* a2= (int*) calloc(n, sizeof(int))
 for (int i= 0; i< n; i++)
    a2[n-1-i]= a1[i];
return(a2);
}

In main
int* res= mirror(a,6)
e poi stampare l'array res con un ciclo for
```

5. 3 punti Su foglio protocollo descrivere le funzioni *malloc*, *calloc*, *realloc*, e *free*, con esempi di codice che le utilizzano.

6. [3 punti] Su foglio protocollo, scrivere una funzione ricorsiva di nome *palindrome* che prende come parametro n un valore di tipo *unsigned int* e stampa, se n uguale a 3, 3210123, richiamando la funzione su $n-1$ fino a 0 (e poi stampando i valori in ordine inverso).

7. [4 punti] Scrivere cosa stampa il seguente programma.

```
1  static int a= -2;
2
3  int f1(int* b, int c) {
4    static int a= 4;
5    if (a <= 6) {
6      //printf("Controllo valori %d %d %d\n", a, *b, c);
7      int d= a++ + ++(*b) + c-- -1 ;
8      printf("%d %d %d %d\n", a, *b, c, d);
9      d= f1(b, c+1);
10     return d;
11   }
12   else {return a;}
13 }
14 int f2(int* b, int c, int* d) {
15   int e= ++a * (*b)++  * c * (*d)++;
16   *d= f1(&c, a);
17   printf("%d\n", a++);
18   return *d*2;
19 }
20 int main(void) {
21   int a= 4, b= 5, c= -2;
22   b= f2(&a, c, &b);
23   printf("%d %d %d\n", a, b, c);
24 }
```

5 -1 -2 1
6 0 -2 3
7 1 -2 5
-1
5 14 -2

8. [2 punti] Su foglio protocollo, scrivere due file, *main.c* e *write.c*: in *main.c* viene dichiarata una variabile a (di tipo *int*) definita in *write.c*, mentre in *write.c* viene definita una funzione di nome *add*, che deve essere dichiarata in *main.c*. La funzione *main* in *main.c* deve leggere da tastiera a e passarlo alla funzione *add*. La funzione *add* ritorna il valore di a incrementato di 1. Infine la funzione *main* stampa il valore di a.

9. [4 punti] Cerchiare le affermazioni vere dato *int a[3]= {65539,258,255}; char *p = (char*) a; short int *q = (short*) a;* sapendo che i tre tipi usati occupano 4, 1, e 2 byte, e $65536 = 2^{16}$ (valori rappresentati in *little endian* e complemento a due). A. $*(p+8)+*(p+5) == -1$; B. $(int)\&a[2] > (int)(q+3)$; C. $(((int)(q+5) - (int)(a+5))\%2)! = 0$; D. $\&a[2] - a >= 1$; E. $q[0] - q[2] >= 8$. F. $*(q+4) - p[4] - p[5] == 252$.

10. [2 punti] Cerchiare le affermazioni vere dato *int a=-1, *const p= &a; const double b= 3.0f, *q= &b; float c= ((a-= !!a, a + 1) >= 0) ? *q*2: 3.0f;* Giustificare le risposte su foglio protocollo. A. Alla fine a vale -2; B. L'inizializzazione di b contiene una conversione di tipo; C. $p = q$ non è permesso; D. $q = NULL$ è permesso; E. L'espressione condizionale ritorna un valore di tipo *double*; F. $\&(\&a)$ genera errore;

Esercizio 9:
vere B, C, D, F, per la correzione vedere correzione esame 3 Febbraio 2017 (a.a. 2016-2017)

Esercizio 10
Vere:
A: !!a vale 1, quindi a-=1 decrementa a a -2
B: 3.0f è una costante letterale di tipo float
C: il puntatore p è un puntatore costante
E: l'espressione può ritornare *q * 2 (double) oppure 3.0f (float), per le regole di conversione di tipo dato che un'espressione può ritornare un valore di un tipo solamente, l'espressione ritorna un double
F: &a è lecito e ritorna l'indirizzo di memoria di dove si trova la variabile a. Questo valore però non è una variabile e quindi &(&a) ritorna un errore (& vuole un lvalue alla sua destra).

ESERCIZIO 2

```
#include<stdio.h>

int isPrimeNumber(int num);
int main() {
  int i, j, rows;
  int counter = 2;

  printf("Enter the number of rows\n");
  scanf("%d", &rows);

  for (i = 1; i <= rows; i++) {
    for (j = 1; j <= i; j++) {
     /* Try to find next prime number by
 incrementing counter and testing it for primality */
      while(!isPrimeNumber(counter)){
         counter++;
 }
      printf("%d ", counter);
      counter++;
    }
    printf("\n");
  }
  return(0);
}

int isPrimeNumber(int num) {
  int i, isPrime = 1;
  for (i = 2; i <= (num/2); i++) {
    if (num % i == 0){
      isPrime = 0;
      break;
    }
  }
  if (isPrime==1 || num==2)
    return 1;
  else
    return 0;
}
```

ESERCIZIO 6

```
void palindrome (unsigned int n) {
 if (n < 0)   {
   printf("Errore");
   return;
 }

if (n == 0) {
   printf("0");
   return;
 }

 printf("%u", n);
 palindrome(n-1)
 printf("%u", n);
}
```

Esercizio 8

main.c

```
extern int a;
int add(void);

int main(void){
 scanf("%d", &a);
 printf("%d",a=add(a));
}
```

write.c

```
int a;
 int add(b){
 return (b+1);
}
```