Prova scritta Programmazione Procedurale

Nome e Cognome: _____ Matricola: _____

1. [1 punti] Evidenziare le conversioni di tipo implicite e scrivere cosa viene stampato a video alla fine, che tipo assume alla fine la variabile *sum* e quanto valgono le variabili x e y alla fine dell'esecuzione del codice.

```
1  int  x = 13;
2  char y = 'C'; /* ASCII value is 67 */
3  float sum;
4  sum = x + y;
5  printf("sum = %f", sum );
6
```

Linea 4, x viene convertito da int a float e y da char a float. Alla fine viene stampato il valore *80.000000* della variabile *sum* che è di tipo float.
Le variabili x e y alla fine dell'esecuzione del codice, rimangono invariate e quindi assumono valore "13" e "C" rispettivamente.

2. [1 punti] Rappresentare i numeri 1, −1 in complemento a due (8bit)

```
1  1 = 00000001
2  -1 = 11111111
3
```

3. [2 punti] Calcolare la seguente somma in complemento a due e scrivere il risultato sia in complemento a due che in base decimale

```
1  00000101 +
2  11110101
```

```
1  00000101 +
2  11110101 =
3  11111010 = -5
4
```

4. [1 punti] Scrivere una funzione che dati in *input* due caratteri qualsiasi, stampi a video il carattere e il relativo valore nella codifica ASCII, con valore ASCII più grande (in caso di uguaglianza si stampi il primo).
Es. *input*: ("*a*", "*h*"), i loro valori nella codifica ASCII sono: 97 e 104, quindi l' *output* dovrà essere del tipo: "Il valore ASCII del carattere *h* è 104".

```
1  void printASCII (char x, char y){
2    if (x >= y){
3      printf("Il valore ASCII del carattere %c
       : %d", x, x);
4    }
5    else {
6      printf("Il valore ASCII del carattere %c
       : %d", y, y);
7    }
8  }
9
```

5. [3 punti] Scrivere una funzione che dati in *input* due numeri interi, calcoli il *massimo comune divisore* di questi.

```
int gcd(int a, int b) {
  int gcd = 0;
  for(int i = a; i >= 1; i--) {
    if(a%i == 0 && b%i == 0){
      gcd = i;
      break;
    }
  }
  return gcd;
}

```

6. 3 punti Scrivere una funzione che richieda di inserire un numero da tastiera e verifichi se questo è *palindromo*, cioè che si legge uguale sia da sinistra verso destra, che da destra verso sinistra (es. 909). Se la condizione è verificata si ritorni il valore intero 1, altrimenti 0.

```
int palindromo() {
  int a, b, c, s = 0;
  printf("Inserisci un numero:\t");
  scanf("%d", &a);
  c = a;
  while(a > 0){
    b = a%10;
    s = (s*10)+b;
    a = a/10;
   }

   if(s == c)
  return 1;
   else
  return 0;
}

```

7. 3 punti Scrivere un programma che dato un array di interi già definito e la sua dimensione (*input*: int *arr*[], int *arr_size*), rimuova i duplicati all'interno di questo e stampi i suoi elementi

```

for(int i = 0; i < arr_size-1; i++){
      for(int j = i+1; j < arr_size; j++){
          if (arr[i] == arr[j]){
              for (int k = j; k < arr_size-1; k++){
                  arr[k] = arr[k+1];
              }
              j--;
              arr_size--;
          }
      }
  }

  for(int i = 0; i < arr_size; i++){
      printf("value%d: %d \n", i+1, arr[i]);
  }

```

8. 2 punti Scrivere una funzione cha dato in input un numero intero, stampi a video un triangolo composto da sequenze di numeri incrementali fino al massimo di n e viceversa. Cioè:

```
Input: 5          Input:3
1                 1
12                12
123               123
1234              12
12345             1
1234
123
12
1
```

```
void triangolo(int n){
  for(int i=1; i <= n*2; i++){
    if (i <=n){
       for(int j = 1; j <= i; j++){
         printf("%d", j);
       }
       printf("\n");
    }
    else{
      for(int j = 1; j <= n*2 - i; j++){
        printf("%d", j);
      }
      printf("\n");
    }
  }
}

```

9. [2 punti] Scrivere una funzione <u>iterativa</u> che, dato in *input* un numero intero, stampi a video il suo valore nella successione di fibonacci

```
int fibonacci_iter(int n){
  if (n == 0){
    return 0;
  }
  if (n == 1 || n == 2){
    return 1;
  }

  int n1 = 1;
  int n2 = 1;
  int fib;
  for (int i = 2; i <= n; i++){
    fib = n1 + n2;
    n1 = n2;
    n2 = fib;
  }
  return fib;
}

```

10. [2 punti] Scrivere una funzione <u>ricorsiva</u> che dato in *input* un numero intero, stampi a video il suo valore nella successione di fibonacci

```
int fibonacci_rec(int n){
  if (n == 0){
    return 0;
  }
  if (n == 1 || n == 2){
    return 1;
  }
  return fibonacci_iter(n-2) + fibonacci_iter
  (n-1);
}

```

11. 1 punti Scrivere una funzione che, dato in *input* un puntatore (di tipo intero) ed un numero intero, calcoli il fattoriale di questo numero e lo salvi nell'area di memoria indirizzata dal puntatore

```
void fattoriale(int *fattoriale, int n){
  if (n == 0){
    *fattoriale = 1;
  }
  else{
    *fattoriale = 1;
    for(int i = 1; i <= n; i++){
      *fattoriale = *fattoriale * i;
    }
  }
}

```

12. 1 punti Definire una struttura per una *linked list* contenente un valore intero e un puntatore al prossimo nodo.

```
typedef struct node {
  int val;
  struct node * next;
} node_t;

```

13. 2 punti Scrivere una funzione che, dato in *input* un puntatore alla testa di una linked list (**head*), iteri e stampi tutti gli elementi della *linked list* sopra definita

```
void print_list(node_t * head) {
  node_t * current = head;

  while (current != NULL) {
    printf("%d\n", current->val);
    current = current->next;
  }
}

```

14. 2 punti Allocare dinamicamente un'area di memoria 10x10 di tipo double

```
double* A = (double*)malloc(10 * 10 * sizeof(
  double));

```

15. [2 punti] Scrivere cosa stampa a video il seguente codice ed indicare i valori di tipo intero, contenuti all'interno dell'area di memoria indirizzata dal puntatore *ptr*. Inoltre scrivere una porzione di codice che permetta la stampa di tutti gli elementi contenuti nell'area di memoria indirizzata dal puntatore *ptr*.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6    int *ptr;
7    ptr = (int *) calloc(5, sizeof(int));
8
9    for (int i = 1; i < 5; i++){
10     *(ptr+i) = i;
11   }
12
13   printf("%d", *ptr);
14   return 0;
15 }
16
```

Stampa a video il numero *0*.

Nell'area di memoria indirizzata da *ptr* sono contenuti i valori: *0, 1, 2, 3, 4*

```
1  for (int i = 1; i < 5; i++){
2        printf("%d \n", *ptr+i);
3  }
4
```