ESAME DI PROGRAMMAZIONE II

1. Date tre liste L1, L2, L3 definire una funzione **iterativa** in C che restituisca in un intero indicante quanti elementi di L3 sono compresi tra il corrispondente elemento di L1 e il corrispondente elemento di L2, quanto L1 o L2 sono terminate si consideri l’elemento/i mancante sostituito dal valore 0. Si discuta la complessita’ in spazio e tempo della funzione. (10 punti)

Esempio: L1: 3 🡪 2🡪 -2 🡪 7

L2: 1 🡪 4🡪 2 🡪 9 🡪 2 🡪 12 🡪 2 🡪16 🡪 7 🡪 2

L3: 5 🡪 6 🡪 -1🡪 8 🡪 -7 🡪 2

OUTPUT: 3

1. Data una lista L1, un intero n, e un intero x (maggiore o uguale a zero), si definisca una funzione **ricorsiva**  in C che restituisca in output il numero di istanze di n presenti dopo la posizione x. Si discuta la complessita’ in spazio e tempo della funzione. (punti 10)

Esempio L1: 1🡪3🡪6🡪7🡪6🡪6🡪8🡪1 🡪1🡪6 🡪 6 🡪 6 🡪 6

n: 6

x : 3

OUTPUT: 6

1. Si descriva l’slgoritmo di QuickSort? Si mostri inoltre perche’ la complessita’ in spazio del QuickSort e’ lineare nel caso peggiore, e logariitmica nel caso migliore. (5 punti)
2. Si simuli l’esecuzione della funzione ricorsiva f, utilizzando i record di attivazione, e supponendo che la funzione venga richiamata con L1: 2🡪2🡪18🡪7 e nella seguente situazione:

x=3; y=1; f(y,1,&x,L1); (8 punti)

f(int a, int b, int \* c, LINK lis)

{ int x=1;

if (lis != NULL)

if (lis->d < (a\*b)

{\*c=(\*c)+x+b; printf(“%d\n”,\*c); f(a,b+1, c, lis->next); printf(“%d\n”,b); printf(“%d\n”,\*c);}

else if (lis->d < (a\*b))

{ printf(“%d\n”,\*c); f(a,b+1,c, lis->next); \*c=(\*c)-x\*b; printf(“%d\n”,b); printf(“%d\n”,\*c); }

}

**Nota**. Si specifichino ***chiaramente*** gli indirizzi di ritorno.