ESAME DI PROGRAMMAZIONE II

1. Date due liste L1 ed L2 definire una funzione **ricorsiva** in C che restituisca in output una nuova lista (si utilizzi la funzione newelement vista a lezione) contenente, posizione per posizione, il prodotto del (contenuto del) nodo di L1e di L2, ma solo se la loro somma e’ multipla della posizione dei nodi. Si discuta la complessita’ in spazio e tempo della funzione. (10 punti)

Esempio: L1: 3 🡪 4🡪 2 🡪 7 🡪 5 🡪 6 🡪 11 🡪16 🡪 7 🡪 2

L2: 0 🡪 2🡪2 🡪 6 🡪 2 🡪 12 🡪 2

OUTPUT: 0🡪8🡪10🡪72

1. Data una lista L1 ed un intero n (0≤n), si definisca una funzione **iterativa** in C che restituisca in output quante un nodo e’ uguale al suo successore, partendo da una posizione maggiore di n. Si discuta la complessita’ in spazio e tempo della funzione. (punti 10)

Esempio L1: 1🡪3🡪6🡪6🡪6🡪6🡪8🡪1 🡪1🡪1 🡪 6 🡪 6 🡪 10

n: 3

OUTPUT: 5

1. Si descriva l’algoritmo di QuickSort? Si mostri inoltre perche’ la complessita’ in spazio del QuickSort e’ lineare nel caso peggiore, e logaritmica 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: 5🡪6🡪2🡪8🡪7 e nella seguente situazione:

x=1; y=0; f(x,1,&y,L1); (8 punti)

void 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.