Giorno 1  
Test- stili di programmazione – Programmazione modulare – Documentazione del codice.  
Pdf o txt- CognomeNome e in caso numero del gruppo es: Appicelorenzo 21  
**Test & Debugging**Test: post codice  
Debugging: durante codice  
Progetto->codifica->Test  
Test cerca gli errori  
Non esiste codice senza errori quindi se non li troviamo abbiamo sbagliato.  
09/03/23  
Tecnica di programmazione per evitare errori: **programmazione difensiva**  
Se una funzione restituisce dei valori es di errori, il valore va verificato con la funzione chiamante.  
Dato un array indicare i divisori comuni dei   
**Dati input**  
Valori, elenco di numeri, array di interi  
DimV, numero di elementi in Valori, intero >0  
**Dati di output**  
DivisoriComuni, insieme contenente i divisori comuni agli elementi di A, array di interi  
**Dati di lavoro**  
F, array di appoggio, array di interi  
Algoritmo  
i=1  
  
Mentre(i<=DimV)  
   
 i=i+1  
Fine  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Giorno2:  
Esame della matrice di capoliste:  
Per ogni giornata calcolare punteggio di ogni squadra fino alla fine della giornata esaminata, trovare il valore massimo dei punteggi calcolati.  
**Input:**  
Risultati, matrice che contiene i risultati di ogni squadra (sulle colonne) di tutte le giornate (le righe) del campionato, i risultato sono i valori 0,1,2 array a 2 dimensioni di interi, valori compresi tra [0,2]  
NumSquadre, numero di squadre che partecipano, intero>1  
Numgiornate,numero di giornate disputate  
**Output**  
LeCapoliste, contiene il nome delle squadre capolista di ogni giornata del campionato, array a 1 dimensione di interi, >0<= NumSquadre  
**Algoritmo:**  
g=1  
Mentre(g<=NumCapoliste)  
 ClassificaGiornata=CalcolareClassificaGiornata(Risultati,NumSquadre,NumGiornate,g)  
 Lecapoliste=TrovareMassimo(ClassificaGiornata,NumSquadra) in posizione g  
 g=g+1  
Fine   
**Funzione TrovareMassimo  
Input**Classifica, punteggi di tutte le squadre, array a 1 dimensione di interi >= 0  
Dimensione, numero di squadre che partecipano al campionato, intero>1  
**Output**LaCapolista, la squadra in testa al campionato,intero>0 <=NumSquadre  
**Algoritmo**  
Max = primo elemento di Classifica  
Capolista = 1  
I=2  
Mentre(2<=Dimensione)  
 Se(elemento di Classifica in posizione I > Max)  
 Allora Max= elemento di Classifica in posizione I  
 Capolista=I  
 Fine  
I=I+1  
Fine  
 **Funzione CalcolareClassificaGiornata  
Input**NumGiornate,  
NumSquadre,  
giornata, la giornata in cui calcolo la classifica intero>0 <=NumGiornata  
**Output**  
ClassificaGiornata, punteggi di tutte le squadre fino alla giornata g, Array a 1 Dim di interi>0 **Algoritmo**  
i = 1  
Mentre(I <NumSquadre)  
 ClassificaGiornata in posizione i = SommarePunti(Risultati,i,g)  
 i=i+1  
Fine  
**Funzione SommarePunti  
Input**Risultati  
Squadra, la squadra di cui voglio calcolare il punteggio,intero>0<=NumSquadre  
Giornata,  
**Output**  
Punti, Punti ottenuti da squadra fino alla giornata giornata, intero >=0<=2\*Giornata  
**Algoritmo**I=1  
Mentre(i<=giornata)  
 Punti=Punti+elemento di Risultati in posizione i e Squadra  
 i=i+1  
Fine  
**Fasi di test:**  
**Test sui valori limite  
Test per classi di equivalenza  
Valori Validi:**  
-Giornata: ]0,NumGiornate] ---> 7  
-Squadre: ]0,NumSquadre] ---> 7  
**Non validi:**  
-Giornata: ]NumGiornate, + infinito[ --> Numgiornate+1  
-Giornata: ]-infinito,0] --> -5  
-Squadre: ]NumSquadre, + infinito[ --> NumSquadre+1  
-Squadre:]-infinito,0] --> -5  
n=0  
n=1  
n= + infinito  
n=12

**Esercizio esame stringa cat in matrice  
Input**Matrice, caratteri qualunque, array a 2 dimensioni di caratteri  
Righe, numero di righe di matrice, intero>0  
Colonne, numero di colonne di matrice, intero>0  
StringadaCercare,stringa da carcare nelle righe e enelle colonne di matrice, array a 1 dimensione di caratteri con finestringa come ultimo carattere  
**Output**  
Occorrenze,numero di volte che la stringa è presente nella matrice, intero>=0  
**Algoritmo**  
Occorrenze=0  
Occorrenze=Occorrenze + RicercaNelleRighe(Matrice,Righe,Colonne,StringaDaCercare)  
Occorrenze= Occorrenze + RicercanelleColonne(Matrice,Righe,Colonne,StringheDaCercare)  
  
RicercaNelleRighei=1  
Mentre(i<=Righe)  
 Occorrenze=Occorrenze+CercareSottoStringa(Matrice,i,StringaDaCercare,Colonne)  
 i=i+1  
 Fine