**Relazione di Progetto IaP**

**Nome gioco**: Il King delle Cartes  
**Componenti:**

* Christian Occhiogrosso
* Leonardo Serpelloni
* Mattia Rosin

**1. Struttura del Progetto**  
La struttura del codice è ordinata e suddivisa in componenti principali:

* **Mazzo di carte**: rappresentato da un array statico di 40 carte. Le carte vengono create, mescolate con l’algoritmo Fisher–Yates e distribuite all'inizio di ogni fase.
* **Giocatori**: contenuti in un array statico di massimo 20 elementi. Ogni giocatore ha un nome, un ID, i punti vita e due carte (una inizialmente coperta). Dopo ogni fase, i giocatori con 0 punti vita vengono eliminati riscrivendo l’array, cioè spostando tutti i giocatori vivi a sinistra.
* **Gestione delle fasi**: ad ogni fase:  
  - il mazzo viene mescolato;  
  - vengono distribuite due carte a ciascun giocatore;  
  - si sceglie casualmente il primo che comincia la fase;  
  - ogni giocatore gioca in ordine ad anello.
* **Effetti delle carte**: implementati esattamente secondo le regole della consegna:  
  - Asso: il giocatore perde 1 vita e lascia la vita sul campo;  
  - Carte 2–6: nessun effetto;  
  - 7: forza il giocatore successivo a scoprire e attivare la carta coperta;  
  - J: dona 1 vita al precedente;  
  - Q: dona 1 vita al secondo successivo (salta uno);  
  - K: raccoglie tutte le vite lasciate sul campo.

Tutti gli effetti sono gestiti in cascata quando necessario (es. un 7 può attivare un Asso altrui). Abbiamo implementato la gestione del numero di giocatori e del numero di punti vita iniziali, impostabili dinamicamente tramite input dell’utente, con un controllo che garantisce che i giocatori rientrano tra i 2 e i 20 e che i punti vita siano maggiori di 2. Dopo ogni fase, viene chiamata “eliminaGiocatori” e viene riscritto l’array riducendo la dimensione se ci sono giocatori eliminati.

**2. Organizzazione del Lavoro**

Il lavoro è stato svolto in modo collaborativo, con una chiara suddivisione dei compiti tra i tre membri del gruppo. Ogni volta che uno di noi incontrava una difficoltà, gli altri erano pronti a intervenire per risolverla insieme, spiegare una funzione, suggerire un’ottimizzazione o semplicemente rivedere insieme una logica non chiara.

* **Christian** si è occupato principalmente della gestione dell’eliminazione dei giocatori, implementando la funzione che riscrive l’array dei giocatori vivi a fine fase, mantenendo l’ordine corretto. Ha inoltre curato in modo dettagliato l’interfaccia utente, lavorando sull’uso dei colori ANSI per rendere l’output leggibile e coinvolgente, e sull’organizzazione del layout testuale con menu e messaggi di input/feedback.
* **Leonardo** ha preso in carico la parte relativa alla gestione dei turni ciclici, curando la logica ad anello che garantisce l’ordine corretto a partire da un giocatore casuale. Ha inoltre lavorato alla struttura generale delle fasi di gioco, contribuendo a integrare correttamente tutte le parti del programma nel ciclo principale.
* **Mattia** ha curato la gestione del mazzo di carte, occupandosi sia della sua creazione secondo le specifiche (valori e semi), sia dell’algoritmo di mescolamento (Fisher–Yates) e della distribuzione delle carte ai giocatori a inizio fase. Inoltre, ha sviluppato la logica di inizializzazione dei giocatori, compresa la raccolta dei nomi e la selezione del numero di partecipanti.

Va comunque sottolineato che, nonostante questa divisione, nessuna parte del codice è stata sviluppata in modo completamente isolato. Abbiamo lavorato a stretto contatto durante tutta la progettazione e implementazione, confrontandoci spesso anche su sezioni non di nostra competenza diretta. Questo ha permesso di migliorare la coerenza del codice e trovare soluzioni più robuste soprattutto nei punti critici, come la gestione degli effetti speciali delle carte (es. attivazioni a catena con il “7”) o l’eliminazione dinamica dei giocatori.

Abbiamo cercato, per quanto possibile, di lasciare autonomia individuale nella scrittura delle parti assegnate, ma sempre pronti ad aiutare quando necessario. Questo approccio ha favorito sia l’efficienza che l’apprendimento reciproco.

**3. Difficoltà Incontrate**  
Durante lo sviluppo del progetto, abbiamo affrontato diverse difficoltà sia tecniche che organizzative. Alcuni aspetti, inizialmente sottovalutati, si sono rivelati più complessi nella fase di implementazione pratica, richiedendo confronto, test, revisioni e talvolta la modifica della struttura del codice già scritto.

* Una delle problematiche principali è stata la **scelta tra array statici e dinamici.** Inizialmente avevamo considerato l’utilizzo di malloc per una gestione più flessibile, ma per evitare problemi di gestione della memoria e complicazioni con i puntatori, abbiamo deciso di utilizzare array statici sia per i giocatori che per il mazzo. Questo ha reso il codice più leggibile e robusto, dato anche il numero massimo fissato di 20 giocatori.
* La **rimozione dei giocatori eliminati** è stata un’altra sfida. Non avendo usato array dinamici né flag per marcare i giocatori “morti”, abbiamo implementato una funzione che riscrive l’array dei giocatori a ogni fine fase, copiando solo quelli con punti vita maggiori di zero nelle prime posizioni. Questa soluzione si è dimostrata efficace e coerente con la scelta di non utilizzare memoria dinamica.
* Un’altra difficoltà significativa è stata la **gestione dell’effetto a catena** della carta “7”, che obbliga a scoprire e applicare l’effetto della carta coperta del giocatore successivo. Inizialmente questo effetto era stato implementato in modo ricorsivo, ma abbiamo riscontrato problemi di controllo del flusso e rischi di loop infiniti. Abbiamo quindi riprogettato l’effetto usando un approccio iterativo, che assegna temporaneamente il turno al giocatore successivo e applica l’effetto senza perdere il controllo della logica del turno corrente.

Dal punto di vista dell’interfaccia utente, la **gestione dell’input** tramite scanf e getchar ha inizialmente creato problemi, soprattutto per via del buffer residuo che impediva l’acquisizione corretta degli input successivi. Abbiamo risolto introducendo una funzione di pulizia del buffer che viene chiamata sistematicamente dopo ogni input da tastiera.

**4. Conclusioni**

Il progetto ci ha permesso di mettere in pratica in modo concreto molte delle nozioni apprese durante il corso, non solo a livello di programmazione in C, ma anche nella gestione del lavoro di gruppo, nella pianificazione e nella risoluzione di problemi complessi.

Dal punto di vista organizzativo, lavorare in gruppo ci ha insegnato quanto sia importante comunicare in modo efficace, dividere i compiti con chiarezza ma restare disponibili ad aiutarsi a vicenda. Il confronto è stato fondamentale per superare ostacoli e migliorare la qualità del lavoro, anche in termini di leggibilità e pulizia del codice.

Siamo complessivamente soddisfatti del risultato finale e crediamo che il progetto rappresenti una buona sintesi del percorso svolto in questo insegnamento, sia dal punto di vista tecnico che collaborativo.