**Demarchi Alessandro 5193343**

**1 Documentazione / Commenti**

1.a Generare la documentazione usando Doxygen e verificare se la documentazione prodotta permette di orientarsi nella struttura del progetto software. Indicare i punti che potrebbero essere migliorati.

1.b Utilizzando la documentazione e i commenti inseriti nei file .ccp e .h verificare se il progetto software aderisce alle specifiche assegnate durante la prima settimana. Indicare le eventuali specifiche non rispettate.

1.c Verificare che il file README introduca correttamente lo scopo del progetto software e che dia sufficienti informazioni per un corretto uso dell’interfaccia a riga di comando. Indicare eventuali mancanze e/o possibili migliorie.

**2 Compilazione e prima sessione di test**

2.a Verificare se è possibile compilare il progetto

2.b Test dell’interfaccia a riga di comando: l’interfaccia funziona correttamente? L’interfaccia è di facile utilizzo? Quali prove sono state eseguite per fare il test?

**3 Seconda sessione di test**

3.a Definire una procedura di test che permetta di identificare possibili bug nel codice prodotto per le singole classi e i singoli metodi di ogni classe

3.b Riportare i risultati del test

**1 Documentazione / Commenti**

1.a La documentazione Doxygen è stata pensata correttamente, ogni file presenta una adeguata descrizione per ogni funzione, e ognuna di esse è inserita in una apposita sezione (constructors/destructor, basic handling…). Con una eccezione, il file main.cpp: infatti, sarebbe corretto, a mio parere, inserire una breve descrizione anche di questo file, che è quello che permette di comprendere il funzionamento di tutto il progetto attraverso il testing. Sempre in questo file i commenti sono pochi, infatti si limitano a descrivere i tre casi di creazione dei poligoni. Un'altra piccola disattenzione è stata posta nella descrizione della funzione Dump() nel file polygon.cpp,

1.b Le specifiche sono state rispettate precisamente: la classe IsoTriangle presenta i parametri base e height nella sezione private, e la funzione GetSide() nella sezione public in maniera tale che possa essere chiamata all’occorrenza ( e non essendo un attributo come base e height ma una funzione non c’è pericolo che il parametro in se venga modificato). La funzione Draw() funziona correttamente e sfrutta a pieno il polimorfismo: è stata definita virtual =0 nella classe base astratta polygon, in maniera che venga obbligatoriamente ridefinita in ogni rispettiva classe derivata.

1.c