TEST 2 - BUG HUNTING

Il test di oggi consisteva nell'esaminre un codice per cercare di capire quale fosse il suo scopo e poi trovare al suo interno potenziali errori di sintassi, logici o di esecuzione in modo da migliorarlo ed renderlo più appetibile per la user experience.

FUNZIONALITA' APPLICAZIONE

Il primo step è stato proprio quello di leggere il codice e capire per cosa fosse stato pensato. Analizzandolo, si capisce che questo avesse come primo input quello di descrivere cosa fosse in grado di fare ovvero effettuare delle operazioni matematiche come moltiplicazione e divisione o di scrivere una riga di caratteri.

```
void menu ()
{
      printf ("Benvenuto, sono un assitente digitale, posso aiutarti a sbrigare alcuni compiti\n");
      printf ("Come posso aiutarti?\n");
      printf ("A >> Moltiplicare due numeri\nB >> Dividere due numeri\nC >> Inserire una stringa\n");

}
```

Screen della parte di codice che descrive quanto detto sopra.

ERRORI DI SINTASSI

Il secondo step è stato invece quello di rileggere con attenzione il codice per trovare gli errori di sintassi presenti e individuare le casistiche non standard che erano state trascurate nel codice e che potessero rappresentare un bug nel software ed essere quindi utilizzati per attaccare il programma stesso.

Gli errori di sintassi che ho notato sono i seguenti:
1- scanf ("%d", &scelta);
//sintassi non valida: avendo utilizzato la variabile di tipo "char" bisognava utilizzare %c
2- short int a,b = 0;
//sintassi non valida: short int è una variabile tipica di C++ quindi non valida in C
3- scanf ("%f", &a);
//sintassi non valida: avendo utilizzato la variabile di tipo int bisognava utilizzare %d
4- short int prodotto = a * b;
//sintassi non valida: short int è una variabile tipica di C++ quindi non valida in C
5- int a,b = 0;
//sintassi non valida in quanto nell'operazione di divisione è sempre meglio assegnare in partenza la variabile float al numeratore e divisore così da non avere problemi in fase di output in caso di numeri con la virgola
6- printf ("Inserisci il denumeratore:");
//errore di sintassi che non avrebbe compromesso il funzionamento del programma ma avrebbe influito sulla user experience
7- int divisione = a % b;
//errore di sintassi in quanto l'operatore "%" serve per ottenere come risultato il resto nell'operazione di divisione e come già detto sopra nel caso di operazione di divisione la variabile deve essere float
8- scanf ("%s", &stringa);
//sintassi non valida: l'operatore & in questo caso non occorre in quanto si ha utilizzato %s ed è il nome dell'array stesso rappresenta un puntatore al suo primo elemento

Tutto il codice con le correzzioni apportate segue in allegato il report.

CASISTICHE NON STANDARD

Le casistiche non standard che non sono state considerate all'interno del codice invece sono:
1- non considerare il caso in cui l'utente vada ad inserire una risposta che non corrisponde con quanto proposto dal menù che appunto non sia ne A, ne B e ne C.
Soluzione: la soluzione che ho adottato è stata quella di andare a coprire ogni altra possibile risposta diversa da A,B e C e nel caso di risposta al di fuori di queste 3 opzioni il codice ripete tramite l'utilizzo del ciclo "do/while" la sua richiesta fino a quando l'utente non inserisce una risposta valida.
Segue screen del codice.

```
22 do{
23 printf ("\nMenù\n");
24 printf ("\nBenvenuto, sono un assitente digitale, posso aiutarti a sbrigare alcuni compiti\n");
25 printf ("\nCome posso aiutarti?\n");
26 printf ("A. Moltiplicare due numeri\n");
27 printf ("B. Dividere due numeri\n");
28 printf ("C. Scrivere una stringa di testo\n");
29 printf ("Cosa scegli umano?\n");
30 scanf ( "%c" , &risp1);
31 getchar();
32 } while (risp1 ≠ 'A' && risp1 ≠ 'B' && risp1 ≠ 'C');
33
```

2- Non considerare nel blocco riguardante l'operazione di divisione l'ipotesi che venga inserito come denominatore il numero 0 e che renda il calcolo impossibile.
Soluzione: la soluzione che ho adottato è stata inserire all'interno del blocco in primis un testo che ricordi all'utente di non immettere lo 0 come denominatore, e anche un ciclo di controllo condizionale (if/else) che vada appunto a filtrare un possibile comportamento sbagliato dell'utente facendo anche in questo caso (tramite l'utilizzo del ciclo "do/while") ripetere il procediemento fino a quando l'utente non inserisca un denominatore diverso da 0.
Segue screen del codice.

```
118 //DIVISIONE
119 if (risp1 == 'B')
120 {
121 printf("\nOk umano, iniziamo\n");
122 do{
123 printf("\nInserisci il numeratore: \n");
124 scanf("%d",&numeratore);
125 printf("\nRicorda che il denominatore deve essere diverso da 0\n");
126 printf("Inserisci il denominatore: \n");
127 scanf("%d",&denominatore);
128 getchar();
129         if(denominatore ≠0)
130         {
131         dividi = numeratore/denominatore;
132         printf("\nIl risultato della divisione è: %f\n",dividi);
133         }
134         else {
135         printf("\nAh sei tosto non puoi mettere 0\n");
136         printf("\nVisto che sei tonto ripeti tutto da 0\n");
137         }
138         }while(denominatore == 0);
139 printf("\nAddio umano\n");
140 }
141 //FINE DIVISIONE
```

3- Non considerare nel blocco riguardante la stringa, l'immissione di più caratteri di quelli previsti dall'array che in questo caso erano 10 caratteri.
Soluzione: la soluzione che ho adottato è stata in primis quella di includere la libreria <string.h> in calce al codice, poi inserire un reminder per l'utente di non digitare una stringa troppo lunga; ho poi aggiunto un ulteriore filtro tramite il comando "strlen" (che va appunto a contare i caratteri immessi nella stringa) abbinato un ciclo di controllo condizionale tramite "if" che va a filtrare un possibile input troppo lungo dell'utente e gli chiede tramite l'utilizzo del ciclo "do/while" di ripetere l'operazione fin quando non inserisce un testo di lunghezza valida.

Segue screen del codice.

```
147 //STRINGA
148 if (risp1 == 'C')
149 {
150
151 printf("\nOk umano, iniziamo\n");
152 do{
153 printf("\nInserisci la tua frase, e mi raccomando che non sia troppo lunga: \n");
154 scanf("%s", string);
155 if (strlen(string)>10){
156 printf("\nma sei proprio un essere umano! Ti ho avvisato di non scrivere troppo!\n");
157 printf("\nTi do una seconda chance: ");
158 }
159 }while(strlen(string)>10);
160 printf("\nbenissimo umano \n");
161 printf("\nAddio umano\n");
162 }
163 //FINE TUTTO
164
165
166 return 0;
167 }
168
```

4- Non considerare di chiedere all'utente che azione volesse fare dopo la prima scelta nel menù potrebbe influenzare negativamente la sua user experience.
Soluzione: la soluzione che ho adottato è stata quella di chiedere all'utente se dopo aver complato la prima azione, volesse compierne un'altra o uscire dalla sua sessione.
Segue screen del codice.

```
36 //MOLTIPLICAZIONE
37 if (risp1 == 'A')
38 {
39 printf("\nOk umano, iniziamo\n");
40 printf("\nInserisci il primo numero: \n");
41 scanf("%d",&num1);
42 getchar();
43 printf("\nInserisci il secondo numero: \n");
44 scanf("%d",&num2);
45 getchar();
46 moltiplica = num1*num2;
47 printf("Il prodotto tra i due numeri é: %d", moltiplica);
48
49
50         printf("\nE ora cosa vuoi fare?\n");
51         printf("\nA. Eseguire un'altra moltiplicazione?\n");
52         printf ("B. Dividere due numeri\n");
53         printf ("C. Scrivere una stringa di testo\n");
54         printf ("D. Esci\n");
55         printf ("Cosa scegli umano?\n");
56         scanf ( "%c" , &risp2);
57         getchar();
58
59         //risp1nellarep1
60         if (risp2 == 'A')
61         {printf("\nOk umano, iniziamo\n");
62         printf("\nInserisci il primo numero: \n");
63         scanf("%d",&num1);
64         getchar();
65         printf("\nInserisci il secondo numero: \n");
66         scanf("%d",&num2);
67         getchar();
68         moltiplica = num1*num2;
69         printf("Il prodotto tra i due numeri é: %d", moltiplica);
70         }
71
72         //risp2nellarep1
73         if (risp2 == 'B')
74         {
75         printf("\nOk umano, iniziamo\n");
76         do{
77         printf("\nInserisci il numeratore: \n");
78         scanf("%d",&numeratore);
79         printf("\nRicorda che il denominatore deve essere diverso da 0\n");
80         printf("Inserisci il denominatore: \n");
```

Tutto il nuovo codice elaborato segue in allegato a questo report.