|  |  |
| --- | --- |
| **Nome e Cognome** |  |
| **Matricola** |  |
| **Corso**  Poli@Home☐ 1(AAAA-BARB)☐ 2(BARC-BOT)☐ 3(BOU-CASA)☐ 4(CASB-CHZ)☐ 5(CIA-COND)☐ 6(CONE-DELR)☐ 7(DELS-FEQ)☐ 8(FER-GEQ)☐ 9(GER-JOZ)☐ 10(JPA-MALI)☐ 11(MALJ-MOD)☐ 12(MOE-PAK)☐ 13(PAL-PORS)☐ 14(PORT-ROQ)☐ 15(ROR-SIGN)☐ 16(SIGO-TRIO)☐ 17(TRIP-ZZZ)☐ 18(Automotive)☐ Solo Prog☐ | |

**Teoria**

**Domanda 1**

|  |  |
| --- | --- |
|  | *Risultato* |
| Dati i seguenti numeri binari in CA2 determinare la loro rappresentazione decimale:  11100001 (8bit)  100001 (6bit)  0011 (4bit) | n1:  n2:  n3: |
| Passaggi | |

**Domanda 2**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Data la seguente tabella di verità:   |  |  |  |  | | --- | --- | --- | --- | | ***a*** | ***b*** | ***c*** | ***f*** | | 0 | 0 | 0 | 1 | | 0 | 0 | 1 | 1 | | 0 | 1 | 0 | 1 | | 0 | 1 | 1 | 1 | | 1 | 0 | 0 | 1 | | 1 | 0 | 1 | 1 | | 1 | 1 | 0 | 1 | | 1 | 1 | 1 | 1 |   Indicare a quale delle seguenti funzioni booleane corrisponde e spiegare:  2.  3. | Funzione:  Spiegazione: |

**Domanda 3**

|  |
| --- |
| Descrivere le principali funzionalità dei registri all’interno della CPU e citare alcuni esempi. |
|  |

**Programmazione**

Un file di testo contiene la mappa di una zona di mare. La mappa è di dimensione **RIGHE**x **COLONNE**, con **RIGHE**e **COLONNE** costanti numeriche definite con due direttive #define; nella mappa i diesis (#) simboleggiano terreno, e i punti (.) mare.

Scrivere un programma per interrogare la mappa. Nel dettaglio, il programma deve leggere la mappa il cui nome è passato come primo argomento dalla linea di comando. Quindi deve permettere all’utente di inserire le coordinate di punti nel formato “**RIGA COLONNA**” e deve verificare:

* Se il punto si trova su un’isola o nel mare
* Se il punto si trova nel mare, identificare la direzione (ALTO, BASSO, DESTRA, SINISTRA) in cui conviene nuotare, sempre nella stessa direzione, per raggiungere più rapidamente un’isola. Il programma deve segnalare se, nuotando nelle direzioni date, non si incontra mai terra e quindi non si trova la minima distanza.

L’inserimento dei punti termina quando l’utente inserisce il punto “**-1 -1**“

Si facciano inoltre le seguenti assunzioni:

* Il contenuto del file è corretto
* Tutte le isole sono rettangolari
* Il punto in alto a sinistra ha coordinate (0, 0)
* L’utente potrebbe inserire una coppia di coordinate non valide (al di fuori della mappa). In questo caso bisogna segnalare l’errore e richiedere nuovamente il dato

## Esempio

Con R=10, C=40. File mappa.dat

........................................

....#####...............................

....#####...............................

....#####...............##############..

....#####.....###.......##############..

....#####.....###.......................

..............###........######.........

..............###........######.........

.........................######.........

........................................

Esecuzione del programma

C:\>**ESAME.EXE mappa.dat**

Inserisci le coordinate (R C): **5 12**

L’isola piu` vicina si trova nuotando verso destra

Inserisci le coordinate (R C): **8 6**

L’isola piu` vicina si trova nuotando verso l’alto

Inserisci le coordinate (R C): **10 0**

Punto fuori dalla mappa. Ripetere.

Inserisci le coordinate (R C): **0 20**

Non e` possibile raggiungere una isola

Inserisci le coordinate (R C): **-1 -1**

Programma terminato

**#include <stdio.h>**

**FILE \*fopen(char \*filename, char \* mode**) – Apertura di un file (mode: “r” lettura – “w” scrittura – “a” append)

**FILE \*freopen(char \*filename, char \* mode, FILE \*file\_pointer**) - Riassegna un file puntatore ad un file diverso.

**int fclose(FILE \*file\_pointer)** - Chiude un file

**int feof(FILE \*file\_pointer) -** Controlla se e' stato incontrato un end-of-file in un file.

**int fflush(FILE \*file\_pointer) -** Svuota il buffer di un file.

**int getchar(void) -** Legge un carattere da "stdin" (tastiera)

**int fgetc(FILE \*file\_pointer) -** Prende un carattere da un file

**char \*gets(char \*buffer) -** Legge una riga da "stdin" (tastiera)

**char \*fgets(char \*string, int maxchar, FILE \*file\_pointer) -** Legge una riga da un file.

**int printf(char \*format \_string, …) -** Scrive output formattato su "stdout" (schermo)

**int fprintf(FILE \*file\_pointer, char \*format\_string, …) -** Scrive output formattato in un file.

**int sprintf(char \*string, char \*format\_string, …) -** Scrive output formattato su una stringa

**int fputc(int c, FILE \*file\_pointer) -** Scrive un carattere in un file

**int putchar(int c) -** Scrive un carattere su "stdout" (schermo)

**int puts(char \*string)** - Scrive una stringa su "stdout" (schermo)

**int fputs(char \*string, FILE \*file\_pointer)** - Scrive una stringa in un file.

**int scanf(char \*format\_string, args)** - Legge input formattato da "stdin" (tastiera)

**int fscanf(FILE \*file\_pointer, char \*format string, args) -** Legge input formattato da file

**int sscanf(char \*buffer, char \*format\_string, args) -** Legge input formattato da una stringa

**EOF** – end of file (costante a valore negativo)

**NULL** - puntatore nullo (valore 0)

**#include <stdlib.h>**

**double atof(char \*string)** - Converte una stringa in un valore in floating point.

**int atoi(char \*string)** - Converte una stringa in un valore integer.

**int atol(char \*string)** - Converte una stringa in un valore long integer.

**void exit(int val)** – Termina il programma, restituendo il valore ‘val’.

**EXIT\_FAILURE** - costante per segnalare terminazione senza successo del programma con exit(); valore diverso da zero

**EXIT\_SUCCESS** - segnala terminazione con successo del programma con exit(); vale 0

**#include <string.h>**

**char \*stpcpy (char \*dest, char \*src) -** Copia una stringa in un'altra. Restituisce dest

**char \*strncpy(char \*s1, char \*s2, size\_t n) -** Copia i primi "n" caratteri di s2 in s1. Restituisce s1

**int strcmp(char \*s1, char \*s2) -** Confronta s1 e s2 per determinare l'ordine alfabetico (<0, s1 prima di s2, 0 uguali, >0 s1 dopo s2)

**int strncmp(char \*s1, char \*s2, size\_t n) -** Confronta i primi "n" caratteri di due stringhe.

**char \*strcpy(char \*s1, char \*s2) -** Copia s2 in s1. Restituisce s1

**int strlen(char \*string) -** Determina la lunghezza di una stringa.

**char \*strcat(char \*s1, char \*s2, size\_t n) -** Aggiunge s2 a s1. Ritorna s1

**char \*strncat(char \*s1, char \*s2, size\_t n) -** Aggiunge "n" caratteri di s2 a s1. Ritorna s1

**char \*strchr(char \*string, int c) -** Cerca la prima occorrenza del carattere ‘c’ in string; restituisce un puntatore alla prima occorrenza di c in s, NULL se non presente

**char \*strrchr(char \*string, int c) -** Cerca l'ultima occorrenza del carattere ‘c’ in string

**char\* strstr(char\* s, char\* t)** - Restituisce un puntatore alla prima occorrenza di t all'interno di s. Restituisce NULL se t non è presente in s.

**char\* strtok(char\* s, const char\* t) -** scompone s in token, i caratteri che delimitano i token sono contenuti in t. Restituisce il puntatore al token (NULL se non ne trova nessuno). Alla prima chiamata in s va inserita la stringa da scomporre e in t i caratteri che delimitano i vari token. Per operare sulla stessa stringa, alle successive chiamate al posto di s si deve passare NULL

**#include <ctype.h>**

**int isalnum(int c)** - Vero se ‘c’ e' alfanumerico.

**int isalpha(int c)** - Vero se ‘c’ e' una lettera dell'alfabeto.

**int iscntrl(int c)** - Vero se ‘c’ e' un carattere di controllo.

**int isdigit(int c)** - Vero se ‘c’ e' un numero decimale.

**int islower(int c)** - Vero se ‘c’ e' una lettera minuscola.

**int isprint(int c)** - Vero se ‘c’ e' un carattere stampabile.

**int ispunct (int c)** - Vero se ‘c’ e' un carattere di punteggiatura.

**int isspace(int c)** - Vero se ‘c’ e' un carattere spazio.

**int isupper(int c)** - Vero se ‘c’ e' una lettera maiuscola.

**tolower(int c)** - Converte ‘c’ in minuscolo.

**int toupper(int c)** - Converte ‘c’ in maiuscolo.

**#include <math.h>**

**int abs (int n) –** valore assoluto intero

**long labs(long n) –** valore assoluto long

**double fabs (double x ) –** valore assoluto di x

**double acos(double x) -** arcocoseno

**double asin(double x) -** arcoseno

**double atan(double x) -** arcotangente

**double atan2(double y, double x) –** arcotangente di y/x.

**double ceil(double x) –** intero superiore a x

**double floor(double x) –** intero inferiore a x.

**double cos(double x) –** x in radianti

**double sin(double x) –** x in radianti

**double tan(double x) –** x in radianti

**double cosh(double x) –** coseno iperbolico

**double sinh(double x) –** seno iperbolico

**double tanh(double x) –** tangente iperbolica

**double exp(double x) -** ex

**double log(double x) -** log(x).

**double log10 (double x ) –** logaritmo base 10

**double pow (double x, double y) -** xy

**int rand (void) –** intero casuale tra 0 e RND\_MAX.

**int random(int max\_num) –** valore casuale tra 0 e max\_num.

**void srand(unsigned seed) –** inizializza la sequenza di valori casuali

**double sqrt(double x) –** radice quadrata

**#include <limits.h>**

**INT\_MAX -** Indica il più grande valore che è possibile rappresentare con un int.

**INT\_MIN -** Indica il più piccolo valore che è possibile rappresentare con un int.

**LONG\_MAX -** Indica il più grande valore che è possibile rappresentare con un long.

**LONG\_MIN -** Indica il più piccolo valore che è possibile rappresentare con un long.

**#include <float.h>**

**FLT\_MAX, DBL\_MAX -** Indica il più grande valore che è possibile rappresentare con un float (o double)

**FLT\_MIN, DBL\_MIN -** Indica il più piccolo valore che è possibile rappresentare con un float (o double)