|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **NOME** |  | | | | **COGNOME** | |  | | | | |
| **MATRICOLA** | | **S** |  |  | |  | |  |  |  | **D/1** |
| **☐A-BARA ☐BARB-BOTS ☐BOTT-CAR ☐CAS-CORD ☐CORE-DIF ☐DIG-FIOR ☐FIOS-GIORD ☐GIORE-LANE ☐LANF-MARA ☐MORB-MOH ☐MOI-PAK ☐PAL-POLH  ☐POLI-ROSA ☐ROSB-SIL ☐SIM-TR ☐TS-ZZZ**  **☐Poli@Home ☐5 Crediti ☐English/A-L ☐English/M-Z ☐Altro:................** | | | | | | | | | | | |

|  |  |
| --- | --- |
| **DOMANDA 1** | *Risultato* |
| Spiegare la principale limitazione della rappresentazione in M&S e motivarla eseguendo la somma su 6 bit tra i seguenti numeri espressi in decimale  n1= -16  n2= -8 | n1(M&S) + n2(M&S)=  limitazione: |
| Passaggi | |

|  |  |
| --- | --- |
| **DOMANDA 2** |  |
| Individuare e commentare l’errore del seguente frammento di codice C  scanf(“%d”, &val);  int v[val];  for(i=0;i<val;i++)  scanf(“%d”, &v[i]); | |
| Risposta | |

|  |  |
| --- | --- |
| **DOMANDA 3** |  |
| Spiegare in cosa consiste la fase di compilazione ed il motivo per cui si rende necessaria. | |
| Risposta | |
| **DOMANDA 4 (PROGRAMMAZIONE)** | |

Si scriva un programma per la gestione degli arrivi ad un porto di N moli. La quantità di moli N è definita tramite una direttiva #define. I moli hanno diverse dimensioni. Le dimensioni (in metri) sono riportate in un file passato come primo argomento da linea di commando, costituito da N righe. Ogni riga del file ha il seguente formato:

**<MOLO> <METRI>**

Dove **MOLO** e **METRI** sono due numeri interi.

Il programma deve gestire l’arrivo di navi di diverse dimensioni e farle attraccare al molo più adatto alle sue dimensioni. Il programma deve chiedere tramite console il nome e la dimensione della prossima nave da far attraccare. Il nome della nave è una stringa di massimo 20 caratteri senza spazi e la dimensione è un intero.

Dopo ogni attracco, il programma deve continuare a chiedere una nuova nave da attraccare, fino al ricevimento della stringa “QUIT” o al momento in cui tutti i moli sono occupati.

Il molo in cui la nave può attraccare non può essere più piccolo della dimensione della nave stessa ma può essere più grande. Tuttavia il programma deve allocare la nave sul molo libero che abbia una lunghezza con minima differenza rispetto a quella della nave. In ogni molo può attraccare una sola nave. In caso non esista alcun molo che possa ospitare la nave il programma deve segnalarlo con il messaggio “**Nessun molo delle dimensioni adatte disponibile: cambiare porto**”.

In caso di moli tutti occupati il programma deve mandare un messaggio su console **“Moli pieni”** e terminare.

Al termine dell’esecuzione il programma deve salvare su un secondo file specificato come secondo argomento sulla linea di comando la situazione degli attracchi per ogni molo, ovvero il numero del molo e il nome della nave attraccata. Non salvare nel file d’uscita le informazioni relative ai moli presso cui non sono presenti navi attraccate.

**Esempio di file moli.txt**

0 60

1 20

2 150

3 30

**Esempio di esecuzione (N=4)**

**#> porto.exe moli.txt attracchi.txt**

**Inserire nome e dimensione nave (in metri):** Tempesta 15

**Attraccare al molo 1**

**Inserire nome e dimensione nave (in metri):** Fortunale 25

**Attraccare al molo 3**

**Inserire nome e dimensione nave (in metri):** Zebra 33

**Attraccare al molo 0**

**Inserire nome e dimensione nave (in metri):** Bolina 170

**Nessun molo delle dimensioni adatte disponibile: cambiare porto**

**Inserire nome e dimensione nave (in metri):** Iceberg 120

**Attraccare al molo 2**

**Moli pieni.**

**Fine programma.**

**File attracchi.txt:**

Molo 0 Zebra

Molo 1 Tempesta

Molo 2 Iceberg

Molo 3 Fortunale

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **NOME** |  | | | | **COGNOME** | |  | | | | |
| **MATRICOLA** | | **S** |  |  | |  | |  |  |  | **D/2** |
| **☐A-BARA ☐BARB-BOTS ☐BOTT-CAR ☐CAS-CORD ☐CORE-DIF ☐DIG-FIOR ☐FIOS-GIORD ☐GIORE-LANE ☐LANF-MARA ☐MORB-MOH ☐MOI-PAK ☐PAL-POLH  ☐POLI-ROSA ☐ROSB-SIL ☐SIM-TR ☐TS-ZZZ**  **☐Poli@Home ☐5 Crediti ☐English/A-L ☐English/M-Z ☐Altro:................** | | | | | | | | | | | |

|  |  |
| --- | --- |
| **DOMANDA 1** | *Risultato* |
| Spiegare la principale limitazione della rappresentazione in M&S e motivarla eseguendo la somma su 7 bit tra i seguenti numeri espressi in decimale:  n1= -32  n2= -16 | n1(M&S) + n2(M&S)=  limitazione: |
| Passaggi | |

|  |  |
| --- | --- |
| **DOMANDA 2** |  |
| Individuare e commentare l’errore del seguente frammento di codice C  int val1, val2;  while (i=0;i<N;i++){  scanf(“%d%d”, &val1, &val2);  printf(“%f”, val1/val2);  } | |
| Risposta | |

|  |  |
| --- | --- |
| **DOMANDA 3** |  |
| Spiegare le principali caratteristiche dei diagrammi di flusso strutturati. | |
| Risposta | |
| **DOMANDA 4 (PROGRAMMAZIONE)** | |

Si scriva un programma per la gestione degli arrivi ad un parcheggio di N piani. La quantità di piani N è definita tramite una direttiva #define. I piani hanno diverso numero di posti complessivi. La quantità di posti per piano sono riportate in un file passato come primo argomento da linea di commando, costituito da N righe. Ogni riga del file ha il seguente formato:

**<PIANO> <POSTI>**

Dove **PIANO** e **POSTI** sono numeri interi.

Il programma deve gestire l’arrivo di macchine e farle parcheggiare al piano più libero, secondo la seguente modalità:

* Il programma deve chiedere all’utente di inserire la targa della macchina da far parcheggiare. La targa è una stringa di 7 caratteri. A parità di minor numero di posti occupati si scelga il primo piano esaminato.
* Tutte le macchine con targhe che cominciano per ZZ vanno fatte parcheggiare al piano 0 indipendentemente dal fatto che sia il più libero o meno. A piano completamente occupato, il programma deve segnalare l’impossibilità di parcheggiare la macchina.
* Dopo ogni inserimento della macchina nel parcheggio, il programma deve continuare a chiedere una nuova macchina da parcheggiare, fino al ricevimento della stringa “QUIT” o al riempimento dell’intero parcheggio.

In caso di parcheggio completamente pieno il programma deve segnalarlo con un messaggio su console e uscire.

Al termine dell’esecuzione il programma deve salvare su un secondo file specificato come secondo argomento sulla linea di comando la situazione dei parcheggi per ogni piano, ovvero il numero del piano e la quantità di parcheggi liberi.

**Esempio di file piani.txt (N=4)**

0 2

1 5

2 5

3 5

**Esempio di esecuzione:**

**#> porto.exe piani.txt parcheggi.txt**

**Inserire targa macchina:** ZZ100BB

**Parcheggiare al piano 0**

**Inserire targa macchina:** BB100CC

**Parcheggiare al piano 1**

**Inserire targa macchina:** CC100DD

**Parcheggiare al piano 2**

**Inserire targa macchina:** ZZ100CC

**Parcheggiare al piano 0**

**Inserire targa macchina:** ZZ200FF

**Parcheggio pieno.**

**Inserire targa macchina:** QUIT

**Fine programma.**

**File parcheggi.txt:**

Piano 0 0

Piano 1 4

Piano 2 4

Piano 3 5