Probleme Examen PC
(Scrieti aici ce v a picat la examen, e ok si cerintele dar daca puneti si rezolvarile, sunteti zei <3)

Prezentarea 1:

P1.1: Se cere de la tastatura o valoare intreaga fara semn pe 32 de biti. Se cere ulterior un index in intervalul [1,24] pozitia 0 fiind bitul LSB. Daca bitul din stanga si cel din dreapta indexului au aceeasi valoare, sa se seteze bitul de la index cu valoarea respectiva. Daca nu, sa se puna bitul de la index paritatea pe biti (0-par, 1-impar). In final se va afisa in hexazecimal si in binar valoarea obtinuta pe intreaga dimensiune.

```
#include <stdio.h>

#include <stdint.h>

void printBits(uint32_t nr)
{
  uint32_t mask = 0x80000000;
  uint8_t i = 0;
  for (i = 0; i < 32; i++)
          {
          if ((nr & mask) == 0)
          {
          printf ("0");
          }
          else
          {
          printf ("1");
          }
          if (i%4 == 3)
          {
          printf (" ");
          }

          mask = mask >> 1;
          }
  printf ("\n");
}

uint32_t index(uint32_t n, int p)
{
  uint8_t mask = 0x01;
  n &= ~(mask << p);
  if(((n >> (p - 1)) & mask) == ((n >> (p + 1)) & mask))
```

```
        {
        n |= ((n >> (p+1)) & mask) << p;
        }
  else
        {
        n |= (((n >> p) & mask) ^ 1) << p;
        }
  return n;
}

int main(void)
{
  uint32_t n = 0;
  printf("n: ");
  scanf("%u", &n);
  printBits(n);

  int p = 0;
  printf("index: ");
  scanf("%d", &p);
  while((p < 1) || (p > 24))
        {
     scanf("%d", &p);
        }

  n = index(n, p);
  printf("%X\n", n);
  printf("0b ");
  printBits(n);
  return 0;
}
```

P1.2: Un sir de caractere care se sfarsea cu “.”. Trebuia sa adaugam “$” in fata fiecarui numar (nu cifra). Spre ex: 123Ana are mere si 23pere. sa devina $123Ana are mere si $23pere.

P1.3: Se da un tablou de numere intregi pe 16 biti fără semn. Sa se implementeze o functie char *arraytostr(uint16_t *array, int size) avand ca parametri tabloul si dimensiunea sa care returneaza un string alocat dinamic care sa contina numerele din tablou scrise în binar, separate prin spațiu. Se va aloca minimul necesar de memorie, iar pentru testare se va defini dimensiunea maxima a tabloului printr-un macro iar tabloul va fi populat cu numere aleatoare din intervalul [0, 500000]. Programul se va termina fara memory leaks.

De exemplu pentru tabloul {12, 200, 123, 1, 700} de dimensiune 5 stringul returnat va fi “0000000000001100 0000000011001000 0000000001111011 0000000000000001 0000001010111100”

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>

#define MAX_SIZE 15


uint16_t *create_array(unsigned int *size) {
    uint16_t *vector = NULL;
    *size = 0;
    uint16_t element;
    while (scanf("%hu", &element) && *size < MAX_SIZE) {
        vector = realloc(vector, (*size + 1) * sizeof(uint16_t));
        if (vector == NULL) {
            perror("Eroare la alocarea dinamica!\n");
            exit(-1);
        }
        vector[*size] = element;
        (*size)++;
    }
    return vector;
}


char *arraytostr(uint16_t *array, int size)
{

    int length = size * (16 + 1) - 1; //avem size numere si fiecare este pe 16 biti, mai adaugam unu pentru ca practic avem 17 biti cu spatiul gol si scadem 1 la final pentru ca avem numerotarea inceputa de la 0
    char *str = malloc((length + 1) * sizeof(char));
    if (str == NULL) {
        perror("Eroare la alocarea dinamica!\n");
        exit(-1);
    }

    char *ptr = str;
    for (int i = 0; i < size; i++) {
        for (int j = 15; j >= 0; j--) {
            *ptr++ = (array[i] & (1 << j)) ? '1' : '0';
        }
        if (i < size - 1) {
            *ptr++ = ' ';
        }
    }
    *ptr = '\0';
    return str;
}
```

```
int main(void) {
    uint16_t *array = NULL;
    char *str = NULL;
    unsigned int size = 0;

    printf("Introduceti numerele (maxim %d):\n", MAX_SIZE);
    array = create_array(&size);
    str = arraytostr(array, size);
    printf("%s\n", str);
    free(array);
    free(str);

    return 0;
}
```

P1.4:Se dau 2 vectori, cu nr necunoscut de elemente. Se face o functie de citire a vectorilor, după aia o funcție de sortare (in ordine crescătoare) a vectorilor. O funcție primește acei 2 vectori și returnează un alt vector alocat dinamic, (tot crescător trebuie sa fie) după următoarea regula: un număr special este acel număr a cărui suma a divizorilor (numărul respectiv nu se pune) este mai mare decât numărul. Vectorul rezultat conține numerele speciale din ambii vectori. Sa se afișeze și dimensiunea vectorului rezultat. (Interclasare pe scurt). Programul se va termina fără memory leaks.
De ex:
pt nr 12: 1+2+3+4+6=16>12=> nr special
v1: 1 2 3 4 6 8 12 20 30
v2: 6 8 12 24 30 36
vectorul rezultat: 12 20 24 30 36
dimensiunea: 5

P1.5: Se da de la tastatură un string de forma “3,12/08/2004” (dw, dd/mm/yyyy) unde dw e ziua săptămânii, dd e ziua lunii, mm e luna și yyyy e anul. Să se scrie o funcție care returneaza un string alocat dinamic folosind doar spatiul minim necesar ca rezultat de forma “miercuri, 12 august 2004”. Se va folosi și o structură corespunzătoare.

P1.6: Se da un string, care contine si numere pana la 6500, oriunde in text. Sa se scrie o functie care returneaza un alt string, alocat dinamic, care inlocuieste numerele, scriindu-le pe 16 biti. (ex: “ana are 17 mere si pune13prune” devine “ana are 0000000000010001 mere si pune0000000000001101prune”)

P1.7: Fie T0=0,T1=1 si T2=1 primii trei termeni ai șirului lui Fibonacci. Sa se scrie o funcție care returnează un SIR DE CARACTERE(contine numerele cu spațiu între ele) alocat dinamic cu primii n termeni ai șirului lui Fibonacci. Antetul funcției: char fibo(int n).
Ex: n=6 se va afișa șirul de caractere
0 1 1 2 3 5 8

P1.8: Sa se scrie o functie inverse_at_bite(?)(uint16_t *array, uint16_t size, uint8_t p, uint8_t n), care primeste ca argumente un array de size elemente, functia complementand fiecare element din array pe urmatorii n biti, incepand de la bitul p.

Sa se aplice pentru un numar nedeterminat de numere(cu malloc). Se poate testa functia cu date introduse de la tastatura sau cu redirectarea intrarii standard, catre fisierul care se poate descarca cu comanda de mai jos:
wget http://staff.cs.upt.ro/~valy/pt/nr.txt.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>

void printBits(uint16_t nr)
{
  uint16_t mask = 0x8000;
  uint8_t i = 0;
  for (i = 0; i < 16; i++)
          {
          if ((nr & mask) == 0)
          {
          printf ("0");
          }
          else
          {
          printf ("1");
          }
          if(i%8 == 7)
        {
          printf(" ");
        }
          mask = mask >> 1;
          }
  printf ("\n");
}

void afisare(uint16_t *v, uint16_t n)
{
  for(int i = 0; i<n; i++)
          {
      printf("%hu ", v[i]);
          }
  printf("\n");
}
```

```
void inverse_at_bite(uint16_t *array, uint16_t size, uint8_t p, uint8_t n)
{
  for(int i = 0; i<size; i++)
          {
      printBits(array[i]);
          for (int j = p; j < n; j++)
        {
          array[i] = array[i] ^ (1<<j);
        }
      printBits(array[i]);
          }
}

int main(void)
{
  uint16_t *array = NULL;
  uint16_t size = 0;
  uint16_t nr = 0;
  while(scanf("%hu", &nr) == 1)
          {
          array = (uint16_t *)realloc(array, sizeof(uint16_t) * (size + 1));
      array[size] = nr;
          size++;
          }

  afisare(array, size);
  inverse_at_bite(array, size, 0, 8);
  free(array);
  return 0;
}
```

P1.9:Se citesc 2 șiruri de caractere se cere sa se facă suma
Ex 7366214790218435

87878
Rezultat 7366214790306313 (tot un string)
Nu se pot transforma numere în int sau long long int (întrucât pot avea și 100 cifre, nu exista o limita)
Trebuie să ținem cont și de cazurile: (în cerință nu a fost menționat dar e logic)
7+8=15 apare o cifra în plus înainte
677+46=723 ținem minte 1 de mai multe ori
9999+1=10000 se schimba cifrele de la capăt ect

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main()
{
   char a[100], b[100], c[100]; // Se declara 3 siruri de caractere pentru a, b si c
   int i, j, k, t, n, m, p;     // Se declara variabilele i, j, k, t, n, m si p
   scanf("%s", a);              // Se citeste primul sir de caractere
   scanf("%s", b);              // Se citeste al doilea sir de caractere
   n = strlen(a);               // Se calculeaza lungimea primului sir
   m = strlen(b);               // Se calculeaza lungimea celui de-al doilea sir
   if (n > m)
   {
       t = n; // Se atribuie t lungimea maxima dintre cele doua siruri
   }
   else
   {
       t = m; // Se atribuie t lungimea maxima dintre cele doua siruri
   }
   p = 0; // Se initializeaza transportul cu 0
   for (i = 0; i < t; i++)
   { // Se parcurg sirurile de la coada la cap
       if (i < n)
       {
           j = a[n - i - 1] - '0'; // Se converteste cifra corespunzatoare din a la
int
       }
       else
       {
           j = 0; // Daca nu exista cifra corespunzatoare in a, se atribuie 0
       }
       if (i < m)
       {
           k = b[m - i - 1] - '0'; // Se converteste cifra corespunzatoare din b la
int
```

```
        }
        else
        {
            k = 0; // Daca nu exista cifra corespunzatoare in b, se atribuie 0
        }
        c[t - i] = (j + k + p) % 10 + '0'; // Se aduna cifrele si transportul si se
stocheaza in c
        p = (j + k + p) / 10;              // Se calculeaza noul transport
    }
    if (p)
    {                       // Daca la final exista transport
        c[0] = p + '0';     // Se adauga transportul la inceputul rezultatului
        printf("%s\n", c); // Se afiseaza rezultatul
    }
    else
    {                           // Altfel
        printf("%s\n", c + 1); // Se afiseaza rezultatul incepand cu cifra de pe
pozitia 1 (fara prima cifra, care este 0)
    }
    return 0; // Se incheie programul
}
```

P1.10: Se da un fisier care contine dictionar de cuvinte de forma cuvant_EN=cuvant_RO. pe fiecare linie. Nu se stie numarul liniilor, dar fiecare are maxim 100 de caractere. Sa se scrie o functie care verifica daca un cuvant in engleza exista in dictionar. Daca da, returneaza traducerea in romana, iar daca nu, returneaza "nu a fost gasit".

Nu se tine cont de literele mari/mici cand se cauta in dictionar.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX 100
#define CHUNK 10

typedef struct
{
  char engleza[MAX];
  char romana[MAX];
}dictionar;

void afisare_struct(dictionar v)
{
  printf("en: %s ro: %s\n", v.engleza, v.romana);
}

void afisare_vector(dictionar *v, int n)
{
  for(int i = 0; i<n; i++)
          {
      afisare_struct(v[i]);
          }
}
```

```
void verificare(dictionar *v, int n, char *cuv)
{
  int flag = 0;
  for(int i = 0; i<n; i++)
          {
    p(cuv, v[i].engleza));
      if(strcasecmp(cuv, v[i].engleza) == 0)
        {  //printf("prelucram cuv %s\n", v[i].engleza);
      //printf("%d\n", strcasecm
          printf("%s\n", v[i].romana);
          flag = 1;
          break;
        }
          }
  if(!flag)
          {
      printf("necunoscut\n");
          }
}

void citire(char *pathname)
{
  FILE *f = NULL;
  if((f = fopen(pathname, "r")) == NULL)
          {
      perror("eroare deschidere");
          exit(-1);
          }

  char buffer[MAX];
  dictionar aux;
  dictionar *v = NULL;
  int i = 0;
  int size_max = 0;
  while(fgets(buffer, MAX, f) != NULL)
          {
          char *p = strtok(buffer, "=");
      strcpy(aux.engleza, p);

          p = strtok(NULL, ".");
      strcpy(aux.romana, p);

          if(i == size_max)
```

```
        {
          v = (dictionar *)realloc(v, sizeof(dictionar) * (size_max + CHUNK));
          size_max += CHUNK;
        }

          v[i] = aux;
      //afisare_struct(v[i]);
          i++;
          }
  //afisare_vector(v, i);
  char cuv[MAX];
  while(fgets(cuv, MAX, stdin) != NULL)
          {
      if(cuv[strlen(cuv) - 1] == '\n')
        {
          cuv[strlen(cuv) - 1] = '\0';
        }
      verificare(v, i, cuv);
          }

  free(v);
  if(fclose(f) != 0)
          {
      perror("eroare inchidere");
          exit(-1);
          }
}

int main(void)
{
  citire("dictionar.txt");
  return 0;
}
```

P1.11:Se citește prin redirectare un fișier care conține perechi de 3 numere în baza 8 să se transforme numere in caractere ascii
Ex 141142143 -> 979899 -> abc
141(baza 8) este 97(baza 10), căruia îi revine caracterul ‘a’

P1.12:Aceeași problema dar citim numere în baza 16, doar ca de aceasta data vom avea perechi de 2 numere.

Ex 616263A06162 -> 979899109798 -> abc\nab adică: abc

61(baza 16) - 97(baza 10) - ‘a’ ab
A0(baza 16) - 10(baza 10) - ‘\n’
Indicații: folosim strtol

P1.13:Sa se creeze 3 funcții(care sa returneze un șir de caractere modificat) si care au ca argument un șir de caractere citit prin redirectare și 1)să elimine vocalele din șir, 2)sa facă toate litere mici in litere mari, restul caracterelor rămânând neschimbate, 3)sa elimine numere formate din mai mult de o cifra. Așa ceva era antetul uneia dintre funcții: char *uppercase(char *s)

P1.14:Sa se implementeze funcția mystrcat care sa respecte documentația din pagina de manual (returnează un pointer la începutul șirului parcă)
Oricum se poate consulta cu man strcat
Char *mystrcat(char *dest,char *src)

P1.15: Sa se scrie o functie set_bits(uint16_t *array, uint32_t size, uint8_t n, uint8_t p), care seteaza n biti de la pozitia p. Se citesc oricate numere de la redirectare sau de la tastatura. Se verifica fiecare argument si se folosesc indecsi. Fisier de test : https://staff.cs.upt.ro/~valy/pt/nr.txt

P1.16: Sa se scrie o functie cu antetul “nume_functie(uint16_t *array, uint32_t size, uint8_t p, uint8_t n)” care complementeaza n biti incepand cu pozitia p, pentru toate numerele din array. Functia va verifica corectitudinea datelor (am intrebat si este ok daca verificam doar daca n si p verifica conditiile de fezabilitate). Se considera ca poate fi un nr nelimitat de numere. Se citeste de la tastatura si prin redirectarea stdin. Se poate testa cu fisierul “nr.txt” (cred).

P1.17: Se citeste de la stdin un nr necunoscut de linii de maxim 300 de caractere pana la linia vida. Se cere sa se ordoneze liniile dupa numarul de vocale si sa se afiseze pe ecran liniile ordonate si nr de vocale din linie.

P1.18: Sa se scrie o functie set_bits(uint16_t array, uint32_t size, uint8_t n, uint8_t p), care seteaza n biti de la pozitia p. Se citesc oricate numere de la redirectare sau de la tastatura. Se verifica fiecare argument si se folosesc indecsi. Fisier de test : https://staff.cs.upt.ro/~valy/pt/nr.txt

P1.19 Sa se aloce dinamic un vector ce contine numere random. Sa se transforme vectorul de tip int intr-un string cu elemente de tip char.

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#define CHUNK 8
#define MAX 100
```

```
int *functie(int *size)
  {
    int aux;
    int *array;
    int index=0;
    int current_size=0;
    current_size=CHUNK;
    srand(time(NULL));
    if((array=malloc(current_size*sizeof(int))) == NULL)
      {
        perror(NULL);
        exit(-1);
      }
    while(1)
      {
        aux=rand();
        array[index++]=aux;
        if(index == current_size)
          {
            current_size=current_size + CHUNK;
            if((array=realloc(array,current_size*sizeof(int))) == NULL)
              {
                perror(NULL);
                exit(-1);
              }
          }
        if(index>=MAX)
          {
            break;
          }
      }
     if((array=realloc(array,current_size*sizeof(int))) == NULL)
              {
                perror(NULL);
                exit(-1);
              }
     *size=index;
     return array;

}

char *array_to_string(int *array, int size)
 {
   char *string;
   int i;
   int offset=0;
   if((string=malloc((size*13+1)*sizeof(char))) == NULL)
```

```
    {
      return NULL;
    }
  for(i=0; i<size; i++)
    {
      offset=offset + sprintf(string+offset,"%d ",array[i]);
    }
  string[offset]='\0';
  return string;
}

int main(void)
{
    int *array=NULL;
    int size=0, i;
    array=functie(&size);
    char *string;
    printf("array:\n");
    for(i=0; i<size; i++)
      {
        printf("%d ", array[i]);
      }
    printf("\n");
    string=array_to_string(array,size);
    printf("string:\n");
    printf("%s ",string);
    free(string);
    free(array);
    return 0;
}
```

P1.20 Se da string-ul alocat dinamic "Acesta este un string si un string se termina in 0x00". Creeati functia:

char *string_replace(char *where, char *what, char *replace)

care modifica sring-ul dat ca intrare(where) inlocuind pe replace in what dupa modelul:

"Acesta este un sir de caractere si un sir de caractere se termina in 0x00"

Problema va fi rezolvata folosind alocarea dinamica si functiile prestabilite din stdlib.h fara memory leaks.

P2.1: Intr-un fișier se scriu datele când concurenții termina o cursa intr-o ordine oarecare. În alt fișier se afișează pe prima linie timpul cel mai rapid în care cineva termina cursa, iar pe restul liniilor diferența de timp mai intai fata de timpul castigator apoi fata de timpul inaintea celui curent.

Ex :

fișier.in

04:05:31
03:02:01
10:18:43

fișier.out
03:02:01
+01:03:30 +01:03:30
+07:16:42 +06:13:12

P2.2: Se da un struct care contine un numar unsigned pe 16 biti si un alt numar intreg. Sa se creeze un vector de struct-uri care parcurge niste valori randomizate. Fiecare struct contine un numar si numarul de aparitii al acelui numar. Sa se scrie in fisierul “out.txt” in format “numar - numar aparitii” ordonate crescator.

P2.3: Se da un fișier text care conține perechi de numere legate prin punct. (de exemplu 123.45). Sa se stocheze aceste perechi de numere intr-un tablou de structuri alocat dinamic, iar perechile se vor ordona crescător (aici perechile de numere legate prin punct se pot interpreta și ca numere reale, gen 312.42, iar prin ordonare crescător se refera ca primul termen al unei perechi trebuie sa fie mai mic decât primul termen al altei perechi iar in cazul in care primul termen al unei perechi este egal cu primul termen al altei perechi, atunci ordonarea se va face dupa al doilea termen, de exemplu 312.42 < 312.99, iar 312.45 < 425.45, exact cum ai ordona numerele reale). Să se scrie aceste numere reale, ordonate crescator intr-un alt fisier pe cate o linie, după următoarea forma nr=[nr]+{nr}, de exemplu:

123.45=123+0.45
312.1=312+0.1
312.42=312+0.42
400.2=400+0.2
412.2=412+0.2

Fișierele de intrare și de iesire se primesc ca argumente in linie de comanda!!!

Ca și fișier de intrare poate fi folosit urmatorul:
http://staff.cs.upt.ro/~valy/pt/points.txt

P2.4: Sa se scrie un program care primeste ca argumente in line de comanda numele a doua fisiere, de intrare si de iesire. Cel de intrare (e recomandat sa fie asta http://staff.cs.upt.ro/~valy/pt/points.txt)

contine coordonate din sistemul xOy sub forma x.y. Se cere sa se scrie numerele in fișierul de iesire ordonate descrescător in funcție de y.Programul se va termina fără memory leaks si se va folosi un tablou de struct alocat dinamic. ok

P2.5: O functie care primeste un string ce contine numere(ex: Ana are 22 de mere, dar 6 pere) si construieste un nou string alocat dinamic care contine propozitia din stringul dat, in care numerele sunt inlocuite cu scrierea lor in hexa.

P2.6: Histograma pentru caracterele litere mici, mari si cifre. Fisierele de intrare/iesire sunt oferite ca argumente in linie de comanda. Se considera ca textul poate fi oricat de lung. Programul va verifica toate cazurile de eroare peosibile. Afisarea se face formatata (caracter - procent% (2 zecimale)).

P2.7: Se da un fisier de intrare si unul de iesire ca argumente in linie de comanda. In fisier sunt triplete de puncte de genul “12.56.101”. Acestea reprezinta coeficientii a b si c de la ecuatia de gradul al 2 lea. Se cere sa se creeze o structura alocata dinamic in care sa se stocheze a b si c, delta, daca are sau nu soltutii reale si x1 si x2. Dupa sa se scrie in fisierul de iesire pentru fiecare triplet:
{a,b,c}-{NU ARE SOLUTII REALE}, daca delta<0
sau
{a,b,c}-{distanta de la orgine la x1, distanta de la origine la x2}, daca delta>=0

P2.8: Se citeste dintr-un fisier csv datele despre cursul euro. Primul camp semnifica data si al doilea camp valoarea in euro. Sa se stocheze dinamic datele intr-o structura cat mai abstract posibila. Programul va folosi argumentele in linie de comanda : ./p fisier.in fisier.out data. Fisier.in contine datele de intrare si va scrie in fisier.out informatia despre cursul euro, din data “data”, din argument in linie de comanda.
Fisierul de test: https://staff.cs.upt.ro/~valy/pt/curs_euro.csv

Ex: 18.01.2022;4.9445
17.01.2022;4.9440
14.01.2022;4.9432
13.01.2022;4.9435
12.01.2022;4.9445
11.01.2022;4.9452

Daca se invoca ./p curs_euro.csv fisier.out 13.01.2022, programul va scrie in fisier.out 13.01.2022;4.9435

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

typedef struct {
    char zi[3];
    char luna[3];
    char an[5];
} Tip_data;

typedef struct {
    Tip_data data;
    char euro[20];
} Tip_baza;

void functie(char *filepath, char *destination, char *thedate) {
    Tip_baza *bd = NULL;
    int dim = 1;
    int nr_date = 0;

    bd = (Tip_baza *)malloc(sizeof(Tip_baza) * dim);

    if (bd == NULL) {
        perror("Eroare la alocare memorie bd");
        exit(EXIT_FAILURE);
    }

    FILE *fis;
    FILE *out;

    if ((fis = fopen(filepath, "r")) == NULL) {
        perror("Eroare la deschidere fis");
        exit(EXIT_FAILURE);
    }

    if ((out = fopen(destination, "w")) == NULL) {
        perror("Eroare la deschidere out");
        exit(EXIT_FAILURE);
    }

    char linie[100];
    char *p = NULL;
    char *q = NULL;
```

```
while (fgets(linie, 100, fis) != NULL) {
   if (linie[strlen(linie) - 1] == '\n')
       linie[strlen(linie) - 1] = '\0';


      char copie[100];
      strcpy(copie,linie);

      strcpy(copie,strchr(copie,';')+1);
   p = strtok(linie, ";");

   q = strtok(p, ".");


   strncpy(bd[nr_date].data.zi, q, sizeof(bd[nr_date].data.zi));

   q = strtok(NULL, ".");
   strncpy(bd[nr_date].data.luna, q, sizeof(bd[nr_date].data.luna));

   q = strtok(NULL, ".");
   strncpy(bd[nr_date].data.an, q, sizeof(bd[nr_date].data.an));


   strncpy(bd[nr_date].euro, copie, sizeof(bd[nr_date].euro));

   nr_date++;

   if (nr_date >= dim) {
      dim = dim * 2;

      bd = (Tip_baza *)realloc(bd, sizeof(Tip_baza) * dim);

      if (bd == NULL) {
         perror("Eroare la realocare memorie bd");
         exit(EXIT_FAILURE);
      }
   }
}

Tip_data sir;
char *w = NULL;

w = strtok(thedate, ".");
strncpy(sir.zi, w, sizeof(sir.zi));
```

```
    w = strtok(NULL, ".");
    strncpy(sir.luna, w, sizeof(sir.luna));

    w = strtok(NULL, ".");
    strncpy(sir.an, w, sizeof(sir.an));

    for (int i = 0; i < nr_date; i++) {
        if (strcmp(bd[i].data.zi, sir.zi) == 0 && strcmp(bd[i].data.luna, sir.luna) == 0 &&
            strcmp(bd[i].data.an, sir.an) == 0) {
            fprintf(out, "%s.%s.%s;%s\n", sir.zi, sir.luna, sir.an, bd[i].euro);
        }
    }

    free(bd);

    if (fclose(fis) != 0) {
        perror("Eroare la inchidere fis");
        exit(EXIT_FAILURE);
    }

    if (fclose(out) != 0) {
        perror("Eroare la inchidere out");
        exit(EXIT_FAILURE);
    }
}

int main(int argc, char *argv[]) {
    char *filepath = NULL;
    char *destination = NULL;
    char thedate[100];

    if (argc != 4) {
        perror("Eroare la argumente");
        exit(EXIT_FAILURE);
    }

    filepath = argv[1];
    destination = argv[2];
    strcpy(thedate, argv[3]);

    functie(filepath, destination, thedate);
    return 0;
}
```

P2.9: Realizeaza o functie care genereaza parole random de o dimensiune data ca argument in linie de comanda. Le genereaza intr-un fisier dat ca argument in linie de comanda.

```
#include <stdio.h>
#include <stdint.h>
#include <string.h>
#include <time.h>
#include <stdlib.h>

void generare(char *destination,char dim[100])
{
  char
alfabet[]="abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456
789";
  int dim1=atoi(dim);
  char parola[dim1];
  int n=0;
   int i;

  srand((unsigned int)(time(NULL)));

    //n=A+rand()%(B-A+1); [A,B]
        for (i = 0; i <dim1; i++)
        {

          n=0+rand()%(strlen(alfabet)-2);


          parola[i]=alfabet[n];

        }

        parola[dim1]='\0';

        FILE *out;

        if((out=fopen(destination,"w")) == NULL)
          {
            perror("Eroare la deschidere out");
            exit(EXIT_FAILURE);
```

```
          }

        fprintf(out,"%s",parola);

        if(!fclose(out))
          {

            perror("Eroare la inchidere out");
            exit(EXIT_FAILURE);
          }
}

int main(int argc,char *argv[])
{
  if(argc !=3 )
    {
      perror("Eroare la argumente");
      exit(EXIT_FAILURE);
    }

    char dim[100];
    char *destination = argv[1];
    strncpy(dim, argv[2], sizeof(dim));
    dim[sizeof(dim) - 1] = '\0';


  generare(argv[1],argv[2]);
  return 0;
}
```

P2.10 Realizati structura unui campionat: Un campionat contine un numar de echipe. Fiecare echipa contine un numar de jucatori, iar fiecare jucator are un id, un nume si un salar. In plus, fiecare campionat si echipa are un nume. Realizati urmatoarele functii: Adaugarea unui jucator in echipa, adaugarea unei echipe in campionat, stergerea unui jucator din echipa, stergerea unei echipe din campionat, afisarea informatiilor despre un jucator, afisarea informatiilor despre o echipa, respectiv afisarea informatiilor despre campionat.

P2.11Un fisier contine pe fiecare linie un numar. Sa se aloce dinamic un vector ce contine toate numerele din fisier si sa se afiseze, sa se ordoneze descrescator si sa se afiseze, apoi sa se afiseze, tot descrescator, fiecare numar scris pe 4 biti. sa se foloseasca argumente in linie de comanda(am uitat in rezolvare)

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#define CHUNK 8
#define MAX 100

void showBits (uint8_t nr)
{
  uint8_t mask = 0x80; // 0b1000000000000000
  uint8_t i = 0;
  for (i = 0; i < 8; i++)
    {

      if ((nr & mask) == 0)
        {
          printf ("0");
        }
      else
        {
          printf ("1");
        }
      mask = mask >> 1; // mask >>= 1;
    }
  printf ("\n");
}

int *functie(int *size)
  {
    FILE *f=NULL;
    if((f=fopen("txt.in","r")) == NULL)
      {
        perror(NULL);
        exit(-1);
      }
    int ch=0;
    int *array;
    int index=0;
    int current_size=0;
    current_size=CHUNK;
    if((array=malloc(current_size*sizeof(int))) == NULL)
      {
        perror(NULL);
        exit(-1);
      }
    while(fscanf(f,"%d",&ch) != EOF)
      {
        array[index++]=ch;
```

```
        if(index == current_size)
         {
           current_size=current_size + CHUNK;
           if((array=realloc(array,current_size*sizeof(int))) == NULL)
            {
              perror(NULL);
              exit(-1);
            }
         }
     }
    if((array=realloc(array,current_size*sizeof(int))) == NULL)
            {
              perror(NULL);
              exit(-1);
            }
    if(fclose(f)!=0)
     {
       perror(NULL);
       exit(-1);
     }
    *size=index;
    return array;
 }

void modificare(int *array, int size)
 {
   int i=0;
   int aux;
   for(i=0; i<size; i++)
    {
      for(int j=i+1; j<size; j++)
       {
         if(array[i]<array[j])
          {
            aux=array[i];
            array[i]=array[j];
            array[j]=aux;
          }
       }
    }
   for(i=0; i<size; i++)
    {
      printf("%d ",array[i]);
    }
   printf("\n");
 }

int main(void)
```

```
{
  int *array=NULL;
  int size=0, i;
  array=functie(&size);
  for(i=0; i<size; i++)
    {
      printf("%d ",array[i]);
    }
  printf("\n");
  modificare(array,size);
  for(i=0; i<size; i++)
    {
      showBits(array[i]);
    }
  printf("\n");
  free(array);
  return 0;
}
```

Prezentarea 2:

P1.1: Se cere citirea de la intrarea standard a unei linii de text de lungime nedefinită și procesarea acesteia prin eliminarea tuturor caracterelor non-litera, cu excepția primului caracter non-litera care apare imediat după un cuvânt
Input:Ioane!!!,Ana are , mere.
Output:Ioane!Ana are mere.
Programul se va termina fara memory leaks.

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>

#define CHUNK 128

char *procesareLinie(void)
{
  char *string = NULL, ch;
  int index = 0, size = 0;

  while((ch = getchar()) != EOF)
  {
    if(ch == '\n')
```

```
    {
      break;
    }
    if(index == size)
    {
      size = size + CHUNK;
      if((string = realloc(string, size * sizeof(char))) == NULL)
      {
        return NULL;
      }
    }
    string[index++] = ch;
  }

  if(string == NULL)
  {
    return NULL;
  }

  if(index == size)
  {
    size = size + 1;
    if((string = realloc(string, size * sizeof(char))) == NULL)
    {
      return NULL;
    }
  }

  string[index] = 0;
  return string;
}

int main(void)
{
  char *linie = NULL;
  if((linie = procesareLinie()) == NULL)
  {
    exit(-1);
  }
  int i, j;
  for(i = 1;i < strlen(linie); i++)
  {
    if((isalpha(linie[i]) == 0) && (isalpha(linie[i-1]) == 0))
    {
```

```
      for(j = i; j < strlen(linie); j++)
      {
        linie[j] = linie[j + 1];
      }
      i--;
    }
  }
  printf("%s\n", linie);
  free(linie);
  return 0;
}
```

P2.1: Se solicită implementarea unei funcții int bin_to_text(char *in, char *out), unde **in** este calea spre un fișier sursă și **out** este calea spre un fișier destinație, ambele specificate ca argumente la linia de comandă. Această funcție va citi secvențial numere de 1 byte din fișierul sursă. Pentru fiecare byte citit, dacă acesta reprezintă o literă (fie majusculă, fie minusculă) sau o cifră, atunci caracterul corespunzător va fi scris în fișierul destinație. Dacă byte-ul citit nu corespunde unei litere sau cifre, funcția va scrie reprezentarea sa hexadecimală, încadrată între paranteze (exemplu: (0E)), în fișierul destinație. Procesul continuă până când toți bytes din fișierul sursă au fost examinați.

Exemple de utilizare:

Pentru un fișier de intrare ce conține bytes 48 65 6C 6C 6F 20 31 21, în fișierul de ieșire se va scrie Hello 31!.
Dacă fișierul de intrare include bytes 7F 41 42 43 30, atunci în fișierul de ieșire se va regăsi (7F)ABC30.

Fisierul de test: https://staff.cs.upt.ro/~valy/pt/integers.bin

Programul se va termina fara memory leaks, etc…