***Cheltuieli de familie***

*Scrieți o aplicație care gestionează cheltuielile de familie efectuate într-o lună. Pentru o cheltuială se vor retine ziua (din luna), suma, tipul cheltuielii (5 categorii: mâncare, întreținere, îmbrăcăminte, telefon, altele). Aplicația să permită efectuarea repetată a următoarelor operații:*

1. *Adaugă cheltuială*

*• adaugă o nouă cheltuială (se specifică ziua, suma, tipul)*

*• actualizează cheltuială (se specifică ziua, suma, tipul)*

1. *Ștergere*

*• Șterge toate cheltuielile pentru ziua data*

*• Șterge cheltuielile pentru un interval de timp (se dă ziua de început și sfârșit, se șterg toate cheltuielile pentru perioada dată)*

*• Șterge toate cheltuielile de un anumit tip*

1. *Căutări.*

*• Tipărește toate cheltuielile mai mari decât o sumă dată •Tipărește toate cheltuielile efectuate înainte de o zi dată și mai mici decât o sumă (se dă suma și ziua, se tipăresc toate cheltuielile mai mici ca suma dată și efectuate înainte de ziua specificată)*

*• tipărește toate cheltuielile de un anumit tip.*

1. *Rapoarte.*

*• Tipărește suma totală pentru un anumit tip de cheltuială*

*•Găsește ziua în care suma cheltuită e maximă*

*• Tipărește toate cheltuielile ce au o anumită sumă*

*• Tipărește cheltuielile sortate după tip*

1. *Filtrare.*

*• Elimină toate cheltuielile de un anumit tip*

*• Elimină toate cheltuielile mai mici decât o sumă data*

1. *Undo*

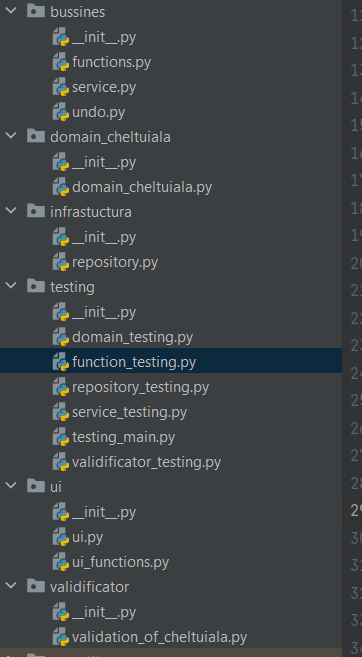
*• Reface ultima operație (lista de cheltuieli revine ce exista înainte de ultima operație care a modificat lista). – Nu folosiți funcția deepCopy*

*Aplicația este bazată pe programare modulară. Pentru a interacționa cu aceasta userul introduce numele comenzii pe care și-o dorește să o execute impreuna cu parametrii ce trebuiesc introdusi.*

*Comenziile pe care userul le poate accesa sunt:*

|  |  |  |
| --- | --- | --- |
| *Comanda user* | *Funcțiile ce sunt apelate* | *Descrierea lor* |
| adauga\_cheltuiala  parametri=zi,tip,suma | ui\_adauga\_cheltuiala | *''' adauga o cheltuiala introdusa de user in lista de cheltuieli* ***:param*** *cheltuieliile: list, lista de cheltuieli* ***:param*** *params: list, in care se afla :ziua cheltuielii-integer, tip-ul cheltuielii-string,suma cheltuielii-float* ***:param*** *stack: list, stiva pentru undo* ***:returvn****: noua lista cu cheltuiala noua '''* |
| actualizeaza\_cheltuiala  parametri= ziua\_de\_modificat, tipul\_de\_modificat,noua\_zi,tip\_nou,suma\_noua | ui\_actualizeaza\_cheltuiala | *''' modifica o cheltuiala existenta in sir . Ea se identifica dupa ziua si tipul ei si userul trebuie sa adauge ziua noua, tipul nou, suma noua* ***:param*** *cheltuieliile: lista de cheltuieli* ***:param*** *params: list, parametrii introdusi de user: zi-int,tip-string al cheltuielii din sir si ziua\_noua-int, tip\_nou-string, suma\_noua-float in care va fi modificat* ***:param*** *stack: stiva pentru undo* ***:return****: lista cheltuieliile cu cheltuiala modificata '''* |
| stergere\_cheltuieli\_zi  parametri=zi | ui\_stergere\_zi | *''' se sterg toate cheltuieliile ce au ziua introdusa de user* ***:param*** *cheltuieliile: list, lista de cheltuieli* ***:param*** *params: list, unde se afla ziua-integer introdusa de user a carui cheltuieli vor fi sterse* ***:param*** *stack: stiva pentru undo* ***:return****: lista modificata '''* |
| stergere\_cheltuieli\_interval\_zilele  parametri= zi\_inferioara si zi\_superioara | ui\_stergere\_interval\_zile | *''' se sterg toate cheltuieliile ce sunt intervalul de zile introdus de user* ***:param*** *cheltuieliile: list, lista de cheltuieli* ***:param*** *params: list, unde se afla zilele-integer, zi\_inf si zi\_sup introduse de user a caror cheltuieli vor fi sterse* ***:param*** *stack: stiva pentru undo* ***:return****: lista modificata '''* |
| elimina\_cheltuieli\_mai\_mici\_ca\_suma  parametri= suma  cu care se va compara | ui\_elimina\_mai\_mici\_ca\_suma | *'''  se sterg toate cheltuieliile ce au suma mai mica ca cea introdusa de user de user* ***:param*** *cheltuieliile: list, lista de cheltuieli* ***:param*** *params: list, unde se afla suma-float introdusa de user a carei cheltuieli vor fi sterse* ***:param*** *stack: stiva pentru undo* ***:return****: lista modificata  '''* |
| elimina\_cheltuieli\_de\_tipul  parametri= tipul | ui\_elimina\_tip | *'''  se sterg toate cheltuieliile ce au tipul introdus de user* ***:param*** *cheltuieliile: list, lista de cheltuieli* ***:param*** *params: list, unde se afla tipul-string introdus de user a carui cheltuieli vor fi sterse* ***:param*** *stack: stiva pentru undo* ***:return****: lista modificata  '''* |
| printeaza\_lista\_cheltuieli | ui\_printeaza\_lista\_cheltuiala | *''' printeaza fiecare cheltuiala pe un rand* ***:param*** *cheltuieliile: list,lista de cheltuieli* ***:param*** *params: -* ***:param*** *stack: stiva pentru undo* ***:return****: printeaza lista de cheltuieli '''* |
| print\_cheltuieli\_mai\_mari\_ca\_suma  parametri= suma | ui\_print\_mai\_mari\_ca\_suma | *''' printeaza cheltuieliile din lista mai mari ca suma introdusa de user* ***:param*** *cheltuieliile: list, lista cheltuieliilor* ***:param*** *params: suma cu care vor fi comparate sumele cheltuieliilor* ***:param*** *stack: stiva pentru undo* ***:return****: printeaza cheltuieliile mai mari ca suma '''* |
| print\_cheltuieli\_mai\_mici\_si\_mai\_devreme  parametri= suma, zi | ui\_print\_mai\_mici\_si\_mai\_devreme | *''' printeaza cheltuieliile mai mici ca o suma si mai devreme precum o data introdusa de user* ***:param*** *cheltuieliile: list, lista de cheltuieli* ***:param*** *params: suma-float, ziua-integer, cu care vor fi comparate cheltuieliile din lista* ***:param*** *stack: stiva pentru undo* ***:return****: printeaza cheltuieliile ce intrunesc conditia '''* |
| print\_cheltuieli\_same\_type  parametri=tip | ui\_print\_same\_type | *''' printeaza cheltuieliile ce au acelasi tip ca tipul introdus de utilizator* ***:param*** *cheltuieliile: list, lista de cheltuieli* ***:param*** *params: tipul-string cu care vor fi comparate cheltuieliile* ***:param*** *stack: stiva de undo* ***:return****: printeaza cheltuieliile de tipul specificat '''* |
| print\_cheltuieli\_suma\_tip\_cheltuiala  parametri=tip | ui\_print\_sum\_same\_type | *''' printeaza suma unui tip specificat de user de la tastatura* ***:param*** *cheltuieliile: list, lista de cheltuieli* ***:param*** *params: tipul- string* ***:param*** *stack: stiva de undo* ***:return****: printeaza toate cheltuieliile ce respecta conditia '''* |
| zi\_cheltuieli\_maxima | ul\_zi\_suma\_maxima | *''' printeaza ziua in care cheltuieliile au cea mai mare suma* ***:param*** *cheltuieliile: list, lista de cheltuieli* ***:param*** *params: -* ***:param*** *stack: stiva de undo* ***:return****: printeaza ziua cu cea mai mare suma '''* |
| print\_aceasi\_sum  parametri=suma | ui\_anumita\_suma | *''' printeaza cheltuieliile ce au aceeasi suma ca suma introdusa de utilizator* ***:param*** *cheltuieliile: list, lista de cheltuieli* ***:param*** *params: suma-float cu care vor fi comparate cheltuieliile* ***:param*** *stack: stiva de undo* ***:return****: printeaza cheltuieliile de suma specificata '''* |
| print\_sortate\_dupa\_tip | ui\_print\_sortate\_dupa\_tip | *''' se creeaza o copie a cheltuieliilor si se sorteaza dupa tip si apoi se printeaza* ***:param*** *cheltuieliile: list, lista de cheltuieli* ***:param*** *params: -* ***:param*** *stack: stiva de undo* ***:return****: printeaza lista sortata '''* |
| help | ui\_print\_menu | *''' daca user-ul are nevoie de ajutor se printeaza un menu pentru al ajuta* ***:param*** *cheltuieliile:list, lista de cheltuieli* ***:param*** *params: -* ***:param*** *stack: stiva de undo* ***:return****: printeaza meniul '''* |
| undo | ui\_undo | *''' functia de refacere a operatiei anterioare Se aplica doar pentru cele ce au un efect asupra listei de cheltuieli nu si la printari* ***:param*** *cheltuieliile: list,lista de cheltuieli* ***:param*** *params: -* ***:param*** *stack: lista acumulata de operatii pentru undo* ***:return****: lista precedenta daca inca se afla ceva in stack  "Nothing to undo" daca nu mai se poate efectua undo '''* |

*Acesta este alcatuit din mai multe packete:*

**

*Scenariu de rulare:*

|  |  |  |
| --- | --- | --- |
| *User-input* | *lista* | *Output* |
| *printeaza\_lista\_cheltuieli* | *[]* | *"Nu exista cheltuieli introduce”* |
| *adauga\_cheltuiala 24 imbracaminte 242.2* | *[{‘zi’: 24, ‘tip’=’imbracaminte’, ‘suma’=242.2}]* | *-* |
|  |  |  |
| *printeaza\_lista\_cheltuieli* | *[{‘zi’: 24, ‘tip’=’imbracaminte’, ‘suma’=242.2}]* | *{‘zi’: 24, ‘tip’=’imbracaminte, ‘suma’=242.2}* |
| *adauga\_cheltuiala 25 imbracaminte 242.2* | *[{‘zi’: 24, ‘tip’=’imbracaminte’, ‘suma’=242.2} ,{‘zi’: 25, ‘tip’=’imbracaminte’, ‘suma’=242.2} ]* |  |
|  |  |  |
| *adauga\_cheltuiala 24 telefon 235.2* | *[{‘zi’: 24, ‘tip’=’imbracaminte’, ‘suma’=242.2} [{‘zi’: 25, ‘tip’=’imbracaminte, ‘suma’=242.2},{‘zi’: 24 ,‘tip’=’telefon’, ‘suma’=235.2} ]* |  |
| *adauga\_cheltuiala 1 mancare 21.32* | *[{‘zi’: 24, ‘tip’=’imbracaminte’, ‘suma’=242.2} [{‘zi’: 25, ‘tip’=’imbracaminte, ‘suma’=242.2},{‘zi’: 24 ,‘tip’=’telefon’, ‘suma’=235.2} , },{‘zi’: 1, ‘tip’=’mancare’, ‘suma’=21.32} ]* |  |
|  |  |  |
| *adauga\_cheltuiala 5 altele 24242.2* | *[{‘zi’: 24, ‘tip’=’imbracaminte’, ‘suma’=242.2} [{‘zi’: 25, ‘tip’=’imbracaminte’,*  *suma’=242.2},{‘zi’: 24 ,‘tip’=’telefon’, ‘suma’=235.2} , },{‘zi’: 1, ‘tip’=’mancare’, ‘suma’=21.32},{‘zi’: 5, ‘tip’=’altele’, ‘suma’=24242.2} ]* |  |
| *printeaza\_lista\_cheltuieli* | *[{‘zi’: 24, ‘tip’=’imbracaminte’, ‘suma’=242.2} [{‘zi’: 25, ‘tip’=’imbracaminte’,*  *suma’=242.2},{‘zi’: 24 ,‘tip’=’telefon’, ‘suma’=235.2} ,{‘zi’: 1, ‘tip’=’mancare’, ‘suma’=21.32},{‘zi’: 5, ‘tip’=’altele’, ‘suma’=24242.2} ]* | *[{‘zi’: 24, ‘tip’=’imbracaminte’, ‘suma’=242.2} [{‘zi’: 25, ‘tip’=’imbracaminte’,*  *suma’=242.2},{‘zi’: 24 ,‘tip’=’telefon’, ‘suma’=235.2} , },{‘zi’: 1, ‘tip’=’mancare’, ‘suma’=21.32},{‘zi’: 5, ‘tip’=’altele’, ‘suma’=24242.2} ]* |
| *stergere\_cheltuieli\_zi 24* | *[{‘zi’: 25, ‘tip’=’imbracaminte’,*  *suma’=242.2},{‘zi’: 1, ‘tip’=’mancare’, ‘suma’=21.32},{‘zi’: 5, ‘tip’=’altele’, ‘suma’=24242.2} ]* |  |
| *elimina\_cheltuieli\_de\_tipul altele* | *[{‘zi’: 25, ‘tip’=’imbracaminte’,*  *suma’=242.2},{‘zi’: 1, ‘tip’=’mancare’, ‘suma’=21.32}]* |  |
| *undo* | *[{‘zi’: 25, ‘tip’=’imbracaminte’,*  *suma’=242.2},{‘zi’: 1, ‘tip’=’mancare’, ‘suma’=21.32},{‘zi’: 5, ‘tip’=’altele’, ‘suma’=24242.2} ]* |  |
| *undo* | *[{‘zi’: 24, ‘tip’=’imbracaminte’, ‘suma’=242.2} [{‘zi’: 25, ‘tip’=’imbracaminte’,*  *suma’=242.2},{‘zi’: 24 ,‘tip’=’telefon’, ‘suma’=235.2} , {‘zi’: 1, ‘tip’=’mancare’, ‘suma’=21.32},{‘zi’: 5, ‘tip’=’altele’, ‘suma’=24242.2} ]* |  |
| *printeaza\_lista\_cheltuieli* | *[{‘zi’: 24, ‘tip’=’imbracaminte’, ‘suma’=242.2} [{‘zi’: 25, ‘tip’=’imbracaminte’,*  *suma’=242.2},{‘zi’: 24 ,‘tip’=’telefon’, ‘suma’=235.2} ,{‘zi’: 1, ‘tip’=’mancare’, ‘suma’=21.32},{‘zi’: 5, ‘tip’=’altele’, ‘suma’=24242.2} ]* | *[{‘zi’: 24, ‘tip’=’imbracaminte’, ‘suma’=242.2} [{‘zi’: 25, ‘tip’=’imbracaminte’,*  *suma’=242.2},{‘zi’: 24 ,‘tip’=’telefon’, ‘suma’=235.2} ,{‘zi’: 1, ‘tip’=’mancare’, ‘suma’=21.32},{‘zi’: 5, ‘tip’=’altele’, ‘suma’=24242.2} ]* |
| *print\_cheltuieli\_mai\_mari\_ca\_suma 236* | *[{‘zi’: 24, ‘tip’=’imbracaminte’, ‘suma’=242.2} [{‘zi’: 25, ‘tip’=’imbracaminte’,*  *suma’=242.2},{‘zi’: 24 ,‘tip’=’telefon’, ‘suma’=235.2} ,{‘zi’: 1, ‘tip’=’mancare’, ‘suma’=21.32},{‘zi’: 5, ‘tip’=’altele’, ‘suma’=24242.2} ]* | *{‘zi’: 24, ‘tip’=’imbracaminte’, ‘suma’=242.2}*  *{‘zi’: 25, ‘tip’=’imbracaminte’,*  *suma’=242.2}*  *{‘zi’: 5, ‘tip’=’altele’, ‘suma’=24242.2}* |
| *adauga\_cheltuiala 40 imbracaminte 24242.2* | *[{‘zi’: 24, ‘tip’=’imbracaminte’, ‘suma’=242.2} [{‘zi’: 25, ‘tip’=’imbracaminte’,*  *suma’=242.2},{‘zi’: 24 ,‘tip’=’telefon’, ‘suma’=235.2} ,{‘zi’: 1, ‘tip’=’mancare’, ‘suma’=21.32},{‘zi’: 5, ‘tip’=’altele’, ‘suma’=24242.2} ]* | *“ziua nu exista!”* |

*For domain:*

|  |  |  |  |
| --- | --- | --- | --- |
| *Tests* | *Detail of test* | | *The function contains* |
| testing\_creeaza() | *''' testeaza daca o cheltuiala de tip dictionar a fost creata corect* ***:return****: valueerror daca nu functioneaza '''* | zi = 23 tip = "mancare" suma = 79.43 cheltuiala = creeaza\_cheltuiala(zi, tip, suma) assert (zi == get\_zi\_cheltuiala(cheltuiala)) assert (tip == get\_tip\_cheltuiala(cheltuiala)) assert (abs(suma - get\_suma\_cheltuiala(cheltuiala)) < 0.00001) | |
| testing\_if\_equal\_even\_if\_cheltuieli\_sum\_dif | *''' testeaza daca doua cheltuieli sunt egale chiar daca au suma diferita* ***:return****: valueerror daca functiile nu sunt bune '''* | zi = 23 tip = "mancare" suma = 79.43 cheltuiala = creeaza\_cheltuiala(zi, tip, suma) assert (zi == get\_zi\_cheltuiala(cheltuiala)) assert (tip == get\_tip\_cheltuiala(cheltuiala)) assert (abs(suma - get\_suma\_cheltuiala(cheltuiala)) < 0.00001) suma\_noua = 84.342 set\_suma\_noua\_cheltuiala(cheltuiala, suma\_noua) assert (abs(suma\_noua - get\_suma\_cheltuiala(cheltuiala)) < 0.00001) cheltuiala\_aceeasi\_zi\_tip = creeaza\_cheltuiala(zi, tip, suma\_noua) assert (egal\_cheltuiala(cheltuiala, cheltuiala\_aceeasi\_zi\_tip)) | |

Repository:

|  |  |  |
| --- | --- | --- |
| Nume functie | *Descriere functie* | Continutul functiei |
| testing\_adaugare\_si\_lungime\_existenta\_in\_lista\_cheltuiala\_corecta() | *''' testeaza daca este adaugata o cheltuiala* ***:return****: -daca cheltuiala seadauga corect '''* | cheltuieliile= [] assert ((numar\_cheltuieli\_lista(cheltuieliile)) == 0) zi = 23 tip = "mancare" suma = 79.43 #o consideram intr-o data corecta cheltuiala = creeaza\_cheltuiala(zi, tip, suma) assert((numar\_cheltuieli\_lista(cheltuieliile))==0) adauga\_cheltuiala\_lista(cheltuieliile,cheltuiala) assert ((numar\_cheltuieli\_lista(cheltuieliile)) == 1) alta\_zi = 24 alt\_tip = "imbracaminte" noua\_cheltuiala = creeaza\_cheltuiala(alta\_zi, alt\_tip, suma) adauga\_cheltuiala\_lista(cheltuieliile, noua\_cheltuiala) assert ((numar\_cheltuieli\_lista(cheltuieliile)) == 2) lista\_cheltuiala = (get\_all\_cheltuieli\_lista(cheltuieliile)) assert (len(lista\_cheltuiala) == 2) assert (egal\_cheltuiala(lista\_cheltuiala[0], cheltuiala)) assert (egal\_cheltuiala(lista\_cheltuiala[1], noua\_cheltuiala)) |
| testing\_adaugare\_aceeasi\_cheltuiala\_in\_lista | *''' Se testeaza daca programul asa cum este si cerut nu lasa utilizatorul sa introduca o aceasi cheltuiala Doua cheltuieli sunt considerate egale daca au aceeasi zi si acelasi tip* ***:return****: - daca se poate adauga cheltuiala  ValueError="cheltuiala existenta" daca sunt egale '''* | cheltuieliile = [] zi = 23 tip = "mancare" suma = 79.43 # o consideram intr-o data corecta cheltuiala = creeaza\_cheltuiala(zi, tip, suma) adauga\_cheltuiala\_lista(cheltuieliile, cheltuiala) same\_zi = 23 same\_tip = "mancare" alta\_suma = 7232.2 alt\_cheltuiala = creeaza\_cheltuiala(same\_zi, same\_tip, alta\_suma)##incerci sa adaugi o cheltuiala cu acceasi chestie try:  adauga\_cheltuiala\_lista(cheltuieliile, alt\_cheltuiala)  assert False except ValueError as ve:  assert (str(ve)=="cheltuiala existenta") |
| testing\_cauta\_cheltuiala\_existenta | *''' se testeaza daca functia din repo cauta corect in lista de cheltuielii o anumita cheltuiala care suntem siguri ca exista* ***:return****: - daca o gaseste  -AssertionError otherwise '''* | cheltuieliile = [] zi = 23 tip = "mancare" suma = 79.43 cheltuiala = creeaza\_cheltuiala(zi, tip, suma) adauga\_cheltuiala\_lista(cheltuieliile, cheltuiala) cheltuiala\_gasita = cauta\_cheltuiala\_dupa\_zi\_tip(cheltuieliile, zi, tip) assert (egal\_cheltuiala(cheltuiala\_gasita, cheltuiala)) |
| testing\_cauta\_cheltuiala\_inexistenta | *''' cauta o cheltuiala de care suntem siguri ca nu exista in sirul de cheltuieli* ***:return****: ValueError="cheltuiala inexistenta\n" daca cheltuiala nu exista in lista '''* | cheltuieliile = [] zi = 23 tip = "mancare" suma = 79.43 cheltuiala = creeaza\_cheltuiala(zi, tip, suma) adauga\_cheltuiala\_lista(cheltuieliile, cheltuiala) alta\_zi = 24 alt\_tip = "imbracaminte" noua\_cheltuiala = creeaza\_cheltuiala(alta\_zi, alt\_tip, suma) try:  cauta\_cheltuiala\_dupa\_zi\_tip(cheltuieliile, alta\_zi, alt\_tip)  assert False except ValueError as ve:  assert (str(ve) == "cheltuiala inexistenta\n") |
| testing\_modifica\_cheltuiala\_inexistenta | *''' se incearca sa se modifice o cheltuiala de care suntem siguri ca nu exista!* ***:return****: ValueError="cheltuiala inexistenta\n" daca se incearca modificare unei cheltuieli inexistente '''* | cheltuieliile = [] zi = 23 tip = "mancare" suma = 79.43 cheltuiala = creeaza\_cheltuiala(zi, tip, suma) adauga\_cheltuiala\_lista(cheltuieliile, cheltuiala) alta\_zi = 24 alt\_tip = "imbracaminte" noua\_cheltuiala = creeaza\_cheltuiala(alta\_zi, alt\_tip, suma) zi\_mod = 25 tip\_mod = "telefon" cheltuiala\_modificata = creeaza\_cheltuiala(zi\_mod, tip\_mod, 231.2) try:  modifica\_cheltuiala\_dupa\_zi\_tip(cheltuieliile,noua\_cheltuiala,cheltuiala\_modificata)  assert False except ValueError as ve:  assert (str(ve) == "cheltuiala inexistenta\n") |
| testing\_sterge\_cheltuiala\_inexistenta | *''' se incearca stergerea unei cheltuielii de care cunoastem ca nu exista in sir* ***:return****: ValueError=="cheltuiala inexistenta\n"  - daca functia nu e corecta '''* | cheltuieliile = [] zi = 23 tip = "mancare" suma = 79.43 cheltuiala = creeaza\_cheltuiala(zi, tip, suma) adauga\_cheltuiala\_lista(cheltuieliile, cheltuiala) alta\_zi = 24 alt\_tip = "imbracaminte" noua\_cheltuiala = creeaza\_cheltuiala(alta\_zi, alt\_tip, suma) try:  sterge\_cheltuiala\_dupa\_zi\_tip(cheltuieliile,alta\_zi,alt\_tip)  assert False except ValueError as ve:  assert (str(ve) == "cheltuiala inexistenta\n") |
| testing\_modificare\_cheltuiala\_existenta() | *''' se testeaza daca functia modificare din repo modifica corect o cheltuiala exista in lista cu alte date introduse* ***:return****: - daca functia a fost implementata corect  AssertionError- in caz contrar '''* | cheltuieliile = [] zi = 23 tip = "mancare" suma = 79.43 cheltuiala = creeaza\_cheltuiala(zi, tip, suma) adauga\_cheltuiala\_lista(cheltuieliile, cheltuiala) zi\_mod = 25 tip\_mod = "telefon" cheltuiala\_modificata = creeaza\_cheltuiala(zi\_mod, tip\_mod, 231.2) modifica\_cheltuiala\_dupa\_zi\_tip(cheltuieliile,cheltuiala, cheltuiala\_modificata) cheltuiala\_gasita = cauta\_cheltuiala\_dupa\_zi\_tip(cheltuieliile, zi\_mod, tip\_mod) assert (get\_tip\_cheltuiala(cheltuiala\_gasita) == tip\_mod and get\_zi\_cheltuiala(cheltuiala\_gasita) == zi\_mod) zin=21 tipn="altele" suman=232.2 zi\_moddd=21 tip\_modd="imbracaminte" suma\_modd=676.2 cheltuiala\_de\_mod=creeaza\_cheltuiala(zin,tipn,suman) adauga\_cheltuiala\_lista(cheltuieliile,cheltuiala\_de\_mod) cheltuiala\_trei=creeaza\_cheltuiala(zi\_moddd,tip\_modd,suma\_modd) modifica\_cheltuiala\_dupa\_zi\_tip(cheltuieliile,cheltuiala\_modificata,cheltuiala\_trei) cheltuiala\_gasita = cauta\_cheltuiala\_dupa\_zi\_tip(cheltuieliile,zi\_moddd, tip\_modd) assert (get\_tip\_cheltuiala(cheltuiala\_gasita) ==tip\_modd and get\_zi\_cheltuiala(cheltuiala\_gasita) == zi\_moddd) |
| testing\_sterge\_cheltuiala\_existenta | *''' se testeaza daca functia de stergere din repo sterge corect o cheltuiala data dupa zi si tip* ***:return****: - daca functia a fost implementata corect  - AssertionError otherwise '''* | cheltuieliile = [] zi = 23 tip = "mancare" suma = 79.43 # o consideram intr-o data corecta cheltuiala = creeaza\_cheltuiala(zi, tip, suma) adauga\_cheltuiala\_lista(cheltuieliile, cheltuiala) alta\_zi = 24 alt\_tip = "imbracaminte" noua\_cheltuiala = creeaza\_cheltuiala(alta\_zi, alt\_tip, suma) adauga\_cheltuiala\_lista(cheltuieliile, noua\_cheltuiala) sterge\_cheltuiala\_dupa\_zi\_tip(cheltuieliile, alta\_zi, alt\_tip) assert (numar\_cheltuieli\_lista(cheltuieliile) == 1) |

*Service:*

|  |  |  |
| --- | --- | --- |
| *Functii* | *Descriere functie* | *Continutul functiei* |
| testing\_adauga\_cheltuiala\_service | *'''  testam daca functia adauga numai in cazuri corecte cheltuiala* ***:return****:- daca cheltuiala valideaza toateconditiile  ValueError- daca zi nu exista, se concateneaza stringul "ziua nu exista\n" la cod  dacat tip-ul nu exista se concateneaza stringul "tipul nu exista\n" la cod  daca pretul <=0, se concateneaza stringul "pretul nu exista\n" la cod  -daca cel putin unul din atributele temei nu este valid:  se arunca exceptie de tipul ValueError cu mesajul codului de eroare de tip string -(l'=lU {cheltuiala} daca nu exista deja in lista o alta cheltuiala cu aceeasi zi si tip  rases ValueError:arunca exceptia de tipul ValueError cu mesajul string  "Cheltuiala invalida"  '''* | cheltuieliile = [] zi = 23 tip = "imbracaminte" suma = 79.43 assert (numar\_cheltuieli\_service(cheltuieliile) == 0) adauga\_cheltuiala\_service(cheltuieliile, zi, tip, suma) assert (numar\_cheltuieli\_service(cheltuieliile) == 1) #verifica daca a crescut try:  adauga\_cheltuiala\_service(cheltuieliile, zi, tip, suma)  assert False except ValueError as ve:  assert (str(ve) == "cheltuiala existenta") zi\_imposibila = 32 tip\_gresit = "factura" suma\_gresita = -54.90 try:  adauga\_cheltuiala\_service(cheltuieliile, zi\_imposibila, tip\_gresit, suma\_gresita)  assert False except ValueError as ve:  assert (str(ve) == "ziua nu exista!\ntipul nu exista!\npretul nu exista!\n") alt\_zi = 22 alt\_tip = "imbracaminte" alta\_suma = 24324.2 lista\_cheltuieli = get\_all\_cheltuieli\_service(cheltuieliile) assert (len(lista\_cheltuieli) == 1) adauga\_cheltuiala\_service(cheltuieliile, alt\_zi, alt\_tip, alta\_suma) assert (numar\_cheltuieli\_service(cheltuieliile) == 2) assert (len(lista\_cheltuieli) == 2) |
| testing\_cauta\_cheltuiala\_service | *''' cauta daca o cheltuiala se afla in sir* ***:return****: cheltuiala daca se afla  else:  cheltuiala gasita daca se afla in sir cautata cu ajutorul unui comutator gasit ce devine pozitia in sir a acesteia  daca este gasita  sau:  raises ValueError("cheltuiala inexistenta\n") in cazul in care cheltuiala nu se afla in sir  se probeaza prin faptul ca gasit va fi -1.'''* | cheltuieliile = [] zi = 23 tip = "imbracaminte" suma = 79.43 adauga\_cheltuiala\_service(cheltuieliile, zi, tip, suma) cheltuiala\_gasita = cauta\_cheltuiala\_dupa\_zi\_tip\_service(cheltuieliile, zi, tip) # verifica daca a fost adaugata assert (get\_zi\_cheltuiala(cheltuiala\_gasita) == zi and get\_tip\_cheltuiala(cheltuiala\_gasita) == tip) alt\_zi = 22 alt\_tip = "imbracaminte" try:  cauta\_cheltuiala\_dupa\_zi\_tip\_service(cheltuieliile, alt\_zi, alt\_tip)  assert False except ValueError as ve:  assert (str(ve) =="cheltuiala inexistenta\n") |
| testing\_modifica\_cheltuiala\_service | *''' se testeaza functia de modificare a unei cheltuieli din service* ***:return****: cheltuiala modificata daca daca se afla in sir cautata cu ajutorul unui comutator gasit ce devine pozitia in sir a acesteia  noua cheltuiala modificata daca aceasta se afla in sir  cautata inca o data prin comutatorul gasit  else:  raises ValueError("cheltuiala inexistenta\n") in cazul in care cheltuiala nu se afla in sir  se probeaza prin faptul ca gasit va fi -1.  raises ValueError("cheltuiala inexistenta\n") - daca cheltuiala nu se afla in sir '''* | cheltuieliile = [] zi = 23 tip = "mancare" suma = 79.43 adauga\_cheltuiala\_service(cheltuieliile, zi, tip, suma) alta\_zi = 24 alt\_tip = "imbracaminte" alta\_suma = 24324.2 suma\_mod=323.2 tip\_mod="altele" zi\_mod=27 try:  modifica\_cheltuiala\_dupa\_zi\_tip\_service(cheltuieliile,alta\_zi,alt\_tip,zi\_mod,tip\_mod,suma\_mod)  assert False except ValueError as ve:  assert(str(ve)=="cheltuiala inexistenta\n") modifica\_cheltuiala\_dupa\_zi\_tip\_service(cheltuieliile, zi, tip, alta\_zi, alt\_tip, alta\_suma) cheltuiala\_gasita = cauta\_cheltuiala\_dupa\_zi\_tip\_service(cheltuieliile, alta\_zi,alt\_tip) assert (get\_zi\_cheltuiala(cheltuiala\_gasita) == alta\_zi and get\_tip\_cheltuiala(cheltuiala\_gasita) == alt\_tip and get\_suma\_cheltuiala(cheltuiala\_gasita)==alta\_suma) |
| testing\_stergere\_cheltuiala\_service | *'''  testeaza daca se sterge o cheltuiala dupa id-ul ei compus din zi si tip* ***:return****: sirul fara cheltuiala daca cheltuiala exista deja in el  else:  raises ValueError deoarece cheltuiala nu exista in sir  '''* | cheltuieliile = [] zi = 23 tip = "mancare" suma = 79.43 adauga\_cheltuiala\_service(cheltuieliile, zi, tip, suma) assert (numar\_cheltuieli\_service(cheltuieliile) == 1) alta\_zi = 24 alt\_tip = "imbracaminte" alta\_suma = 24324.2 try:  sterge\_cheltuiala\_dupa\_zi\_tip\_service(cheltuieliile, alta\_zi,alt\_tip)  assert False except ValueError as ve:  assert (str(ve) == "cheltuiala inexistenta\n") lista\_cheltuieli = get\_all\_cheltuieli\_service(cheltuieliile) adauga\_cheltuiala\_service(cheltuieliile,alta\_zi,alt\_tip,alta\_suma) assert (len(lista\_cheltuieli)== 2) sterge\_cheltuiala\_dupa\_zi\_tip\_service(cheltuieliile, alta\_zi, alt\_tip) assert (len(lista\_cheltuieli) == 1) |

*Validificator:*

|  |  |  |
| --- | --- | --- |
| *Nume functie* | *Detalii functie* | *Continut functie* |
| test\_cheltuiala\_gresita | *''' testeaza validarea unei cheltuielii. Se incearca validarea unei cheltuielii greite* ***:return****: - daca functia ruleaza corect  ValueError "ziua nu exista!\ntipul nu exista!\npretul nu exista!\n") '''* | zi = 23 tip = "mancare" suma = 79.43 cheltuiala = creeaza\_cheltuiala(zi, tip, suma) valideaza\_cheltuiala(cheltuiala) zi\_imposibila = 32 tip\_gresit = "factura" suma\_gresita = -54.90 cheltuiala\_gresita = creeaza\_cheltuiala(zi\_imposibila, tip\_gresit, suma\_gresita) try:  valideaza\_cheltuiala(cheltuiala\_gresita)  assert False except ValueError as ve:  assert (str(ve) == "ziua nu exista!\ntipul nu exista!\npretul nu exista!\n") |