Principles of Programming Language

Assignment 4

Student Details

Name: Krunal Rank
Adm No: U18CO081

1

```
create_list(Nums) :-
   read(Num),
   number(Num),
   create_list(Num,Nums).

create_list(-1,[]) :-
   !.

create_list(Num,[Num|Nums]) :-
   read(Num1),
   create_list(Num1,Nums).

main:- nl,
   write('Enter a Number (-1 to stop) : '),nl,
   create_list(Nums),
   write('Your List : '),
   write(Nums).
```

```
?- main.

Enter a Number (# to stop) :
|: 2.
|: 3.
|: 123.
|: -1.
Your List : [2,3,123]
true.
```

2

```
create_list(Nums) :-
   read(Num),
   number(Num),
```

```
   create_list(Num,Nums).

create_list(-1,[]) :-
   !.

create_list(Num,[Num|Nums]) :-
   read(Num1),
   create_list(Num1,Nums).

list_sum([Item], Item).
list_sum([Item1,Item2 | Tail], Total) :-
   Sum is Item1 + Item2,
   list_sum([Sum|Tail], Total).

main:- nl,
   write('Enter a Number (-1 to stop) : '),nl,
   create_list(Nums),
   write('Your List : '),
   write(Nums),
   nl,
   write('Sum of List Elements : '),
   list_sum(Nums,Sum),
   write(Sum).
```

```
[1]  ?- main.

Enter a Number (-1 to stop) :
|: 1.
|: 2.
|: 3.
|: -1
|: .
Your List : [1,2,3]
Sum of List Elements : 6
true .

[1]  ?- main.

Enter a Number (-1 to stop) :
|: -100.
|: 123213.
|: 2301.
|: -1
|: .
Your List : [-100,123213,2301]
Sum of List Elements : 125414
true .
```

3

```
create_list(Nums) :-
   read(Num),
   number(Num),
   create_list(Num,Nums).

create_list(-1,[]) :-
   !.

create_list(Num,[Num|Nums]) :-
   read(Num1),
   create_list(Num1,Nums).

list_size([],0).
list_size([_|Xs],L) :-
   list_size(Xs,N),
   L is N+1 .

main:- nl,
   write('Enter a Number (-1 to stop) : '),nl,
   create_list(Nums),
   write('Your List : '),
   write(Nums),
   nl,
   write('Size of List : '),
   list_size(Nums,Size),
   write(Size).
```

```
?- main.

Enter a Number (-1 to stop) :
|: 1.
|: 2.
|: 3.
|: -1.
Your List : [1,2,3]
Size of List : 3
true.
```

4.

```
create_list(L):-
   read(C),
   create_list(C,L),
   is_alpha(C).
```

```
create_list(-1,[]) :- !.

create_list(C,[C | L]) :-
   read(C1),
   create_list(C1,L),
   is_alpha(C).

vowel(a).
vowel(e).
vowel(i).
vowel(o).
vowel(u).

count_vowels([], 0).

count_vowels([X|T], N):-
   vowel(X),
   count_vowels(T,N1),
   N is N1+1.

count_vowels([X|T], N):-
   count_vowels(T,N).

main:- nl,
   write('Enter a Letter (-1 to stop) : '),nl,
   create_list(L),
   write('Your List : '),
   write(L),
   nl,
   write('No. of Vowels : '),
   count_vowels(L,Count),
   write(Count),
   nl.
```

```
?- main.

Enter a Letter (-1 to stop) :
|: a.
|: b.
|: c.
|: -1.
Your List : [a,b,c]
No. of Vowels : 1
true .
```

5

```
create_list(Nums) :-
   read_num(Num),
   create_list(Num,Nums).

create_list(-1,[]) :-
   !.

create_list(Num,[Num|Nums]) :-
   read(Num1),
   create_list(Num1,Nums).

read_num(Num) :-
   read(Num),
   number(Num).

main:- nl,
   write('Enter a Number (-1 to stop) : '),nl,
   create_list(Nums),
   write('Enter a Number to search for : '),nl,
   read_num(S),
   write('Your List : '),
   write(Nums),
   nl,
   write('Your Target : '),
   write(S),
   nl,
   member(S,Nums),
   nl.
```

```
?- main.

Enter a Number (-1 to stop) :
|: 1.
|: 2.
|: 3.
|: -1.
Enter a Number to search for :
|: 1.
Your List : [1,2,3]
Your Target : 1

true .
```

```
?- main.

Enter a Number (-1 to stop) :
|: 1.
|: 2.
|: 3.
|: -1.
Enter a Number to search for :
|: 4.
Your List : [1,2,3]
Your Target : 4
false.
```

6

```
create_list(Nums) :-
   read_num(Num),
   create_list(Num,Nums).

create_list(-1,[]) :-
   !.

create_list(Num,[Num|Nums]) :-
   read(Num1),
   create_list(Num1,Nums).

read_num(Num) :-
   read(Num),
   number(Num).

reverse([],Z,Z).

reverse([H|T],Z,Acc) :- reverse(T,Z,[H|Acc]).

main:- nl,
   write('Enter a Number (-1 to stop) : '),nl,
   create_list(Nums),
   write('Your List : '),
   write(Nums),
   nl,
   write('Reversed List : '),
   reverse(Nums,X,[]),
   write(X),
   nl.
```

```
?- main.

Enter a Number (-1 to stop) :
|: 1.
|: 2.
|: 3.
|: -1.
Your List : [1,2,3]
Reversed List : [3,2,1]
true.
```

7

```
create_list(Nums) :-
   read_num(Num),
   create_list(Num,Nums).

create_list(-1,[]) :-
   !.

create_list(Num,[Num|Nums]) :-
   read(Num1),
   create_list(Num1,Nums).

read_num(Num) :-
   read(Num),
   number(Num).

concatenate(List1, List2, Result):-
  append(List1, List2, Result).

main:- nl,
   write('Enter a Number for List 1 (-1 to stop) : '),nl,
   create_list(L1),
   write('Enter a Number for List 2 (-1 to stop) : '),nl,
   create_list(L2),
   write('Your List 1 : '),
   write(L1),
   nl,
   write('Your List 2 : '),
   write(L2),
   nl,
   write('Concatenated List : '),
   concatenate(L1,L2,L),
   write(L),
   nl.
```

```
?- main.

Enter a Number for List 1 (-1 to stop) :
|: 1.
|: 2.
|: -1.
Enter a Number for List 2 (-1 to stop) :
|: 3.
|: 4.
|: -1.
Your List 1 : [1,2]
Your List 2 : [3,4]
Concatenated List : [1,2,3,4]
true.
```

8

```
create_list(Nums) :-
   read_num(Num),
   create_list(Num,Nums).

create_list(-1,[]) :-
   !.

create_list(Num,[Num|Nums]) :-
   read(Num1),
   create_list(Num1,Nums).

read_num(Num) :-
   read(Num),
   number(Num).

remove(S,[S | T],T).

remove(S,[A,B|C],[B|E]) :-
   remove(S,[B|C],E).



main:- nl,
   write('Enter a Number (-1 to stop) : '),nl,
   create_list(Nums),
   write('Enter a Number to delete : '),nl,
   read_num(S),
   write('Your List : '),
   write(Nums),
   nl,
   write('Your Target : '),
   write(S),
```

```
    nl,
    remove(S,Nums,L),
    write('New List : '),
    write(L),
    nl.
```

```
?- main.

Enter a Number (-1 to stop) :
|: 1.
|: 2.
|: 3.
|: -1.
Enter a Number to delete :
|: 1.
Your List : [1,2,3]
Your Target : 1
New List : [2,3]
true .
```

9

```
create_list(Nums) :-
    read_num(Num),
    create_list(Num,Nums).

create_list(-1,[]) :-
    !.

create_list(Num,[Num|Nums]) :-
    read(Num1),
    create_list(Num1,Nums).

read_num(Num) :-
    read(Num),
    number(Num).

maximum_no([X],X).
maximum_no([H|T],X):-
    maximum_no(T,X),
    H @< X.
maximum_no([H|T],H):-
    maximum_no(T,M),
    M @< H.

minimum_no([X],X).
```

```
minimum_no([H|T],X):-
   minimum_no(T,X),
   X @< H.
minimum_no([H|T],H):-
   minimum_no(T,M),
   H @< M.



main:- nl,
   write('Enter a Number (-1 to stop) : '),nl,
   create_list(Nums),
   write('Your List : '),
   write(Nums),
   nl,
   maximum_no(Nums,Max),
   write('Maximum : '),
   write(Max),
   nl,
   minimum_no(Nums,Min),
   write('Minimum : '),
   write(Min),
   nl.
```

```
?- main.

Enter a Number (-1 to stop) :
|: 1.
|: 2.
|: 3.
|: -1.
Your List : [1,2,3]
Maximum : 3
Minimum : 1
true .
```

10

```
create_list(Nums) :-
   read_num(Num),
   create_list(Num,Nums).

create_list(-1,[]) :-
   !.

create_list(Num,[Num|Nums]) :-
   read(Num1),
   create_list(Num1,Nums).

read_num(Num) :-
```

```
   read(Num),
   number(Num).

concatenate(List1, List2, Result):-
  append(List1, List2, Result).

min([H], H, []).
min([H|L], M, [H|R]) :- min(L, M, R), H >= M.
min([H|L], H, [M|R]) :- min(L, M, R), H < M.

sorted([], []).
sorted(L, [M|S]) :- min(L, M, R), sorted(R, S).

main:- nl,
   write('Enter a Number for List 1 (-1 to stop) : '),nl,
   create_list(L1),
   write('Enter a Number for List 2 (-1 to stop) : '),nl,
   create_list(L2),
   write('Your List 1 : '),
   write(L1),
   nl,
   write('Your List 2 : '),
   write(L2),
   nl,
   write('Merged and sorted List : '),
   concatenate(L1,L2,L),
   sorted(L,M),
   write(M),
   nl.
```

```
?- main.

Enter a Number for List 1 (-1 to stop) :
|: 1.
|: 7.
|: 4.
|: -1.
Enter a Number for List 2 (-1 to stop) :
|: 6.
|: 2.
|: 3.
|: 0.
|: -1.
Your List 1 : [1,7,4]
Your List 2 : [6,2,3,0]
Merged and sorted List : [0,1,2,3,4,6,7]
true .
```

```
create_list(Nums) :-
   read_num(Num),
   create_list(Num,Nums).

create_list(-1,[]) :-
   !.

create_list(Num,[Num|Nums]) :-
   read(Num1),
   create_list(Num1,Nums).

read_num(Num) :-
   read(Num),
   number(Num).

pal([]).
pal([_]).
pal(Pal) :-
   append([H|T], [H], Pal),
   pal(T).

main:- nl,
   write('Enter a Number (-1 to stop) : '),nl,
   create_list(Nums),
   write('Your List : '),
   write(Nums),
   nl,
   pal(Nums),
   nl.
```

```
[1]  ?- main.

Enter a Number (-1 to stop) :
|: 1.
|: 2.
|: 1.
|: -1.
Your List : [1,2,1]

true .

[1]  ?- main.

Enter a Number (-1 to stop) :
|: 1.
|: 2.
|: 3.
|: -1.
Your List : [1,2,3]
false.
```

12

```
create_list(Nums) :-
   read_num(Num),
   create_list(Num,Nums),
   number(Num).

create_list(-1,[]) :-
   !.

create_list(Num,[Num|Nums]) :-
   read(Num1),
   create_list(Num1,Nums),
   number(Num).

read_num(Num) :-
   read(Num),
   number(Num).



main:- nl,
   write('Enter a Number (-1 to stop) : '),nl,
   create_list(Nums),
   write('Enter position : '),nl,
   read_num(S),
   write('Your List : '),
   write(Nums),
   nl,
```

```
   write('Required Position : '),
   write(S),
   nl,
   write('Required Element : '),
   nth0(S,Nums,X),
   write(X),
   nl.
```

```
?- main.

Enter a Number (-1 to stop) :
|: 1.
|: 2.3.
|: 1213.
|: 4.
|: -1.
Enter position :
|: 2.
Your List : [1,2.3,1213,4]
Required Position : 2
Required Element : 1213
true.
```

13

```
create_list(Nums) :-
   read_num(Num),
   create_list(Num,Nums).

create_list(-1,[]) :-
   !.

create_list(Num,[Num|Nums]) :-
   read(Num1),
   create_list(Num1,Nums).

read_num(Num) :-
   read(Num),
   number(Num).

product([X],X).

product([H|T],X) :-
   product(T,Y),
   X is H*Y.

main:- nl,
   write('Enter a Number (-1 to stop) : '),nl,
```

```
   create_list(Nums),
   write('Your List : '),
   write(Nums),
   nl,
   write('Product of Elements : '),
   product(Nums,X),
   write(X),
   nl.
```

```
?- main.

Enter a Number (-1 to stop) :
|: 2323.
|: 123.
|: 123.
|: -1.
Your List : [2323,123,123]
Product of Elements : 35144667
true .

?- main.

Enter a Number (-1 to stop) :
|: 1.
|: 2.
|: 3.
|: -1.
Your List : [1,2,3]
Product of Elements : 6
true .
```

14

```
create_list(Nums) :-
   read_num(Num),
   create_list(Num,Nums).

create_list(-1,[]) :-
   !.

create_list(Num,[Num|Nums]) :-
   read(Num1),
   create_list(Num1,Nums).

read_num(Num) :-
   read(Num),
   number(Num).

evenNumbers([],[]).
evenNumbers([H|T],L1):-
```

```
   integer(H),
   (H mod 2 =:=0 -> L1=[H|T1],evenNumbers(T,T1);
   evenNumbers(T,L1) ).

oddNumbers([],[]).
oddNumbers([H|T],L1):-
   integer(H),
   (H mod 2 =:=1 -> L1=[H|T1],oddNumbers(T,T1);
   oddNumbers(T,L1) ).


main:- nl,
   write('Enter a Number (-1 to stop) : '),nl,
   create_list(Nums),
   write('Your List : '),
   write(Nums),
   nl,
   evenNumbers(Nums,E),
   oddNumbers(Nums,O),
   write('Even Elements : '),
   write(E),
   nl,
   write('Odd Elements : '),
   write(O),
   nl.
```

```
?- main.

Enter a Number (-1 to stop) :
|: 1.
|: 2.
|: 3.
|: -1.
Your List : [1,2,3]
Even Elements : [2]
Odd Elements : [1,3]
true.
```