```
(* Γ[c_,a_,b_,met_,coord_] gives Subsuperscript[Γ, ab, c]
 for a given metric (nxn matrix) and coordinate system *)
Γ[c_, a_, b_, met_, coord_] :=
 Module[{imet, n}, n = Length[coord]; imet = Inverse[met];
Return[(1 / 2) Sum[imet[[c, d]] (D[met[[a, d]], coord[[b]]] + D[met[[b, d]],
         coord[[a]]] - D[met[[a, b]], coord[[d]]]), {d, 1, n}]]]
Γ[met_, coord_] := Table[Γ[c, a, b, met, coord], {c, 1, Length[coord]},
  {a, 1, Length[coord]}, {b, 1, Length[coord]}]
```

```
(*This function will take a covariant derivative of an arbitrary tensor field
The result is of the form ∇varTdown^up where
 down and up are lists of particular index values
met is the metric and coord is a list of your symbolic coordinates.*)
CovariantD[T_, up_, down_, var_, met_, coord_] :=
Module[{n, indexList, S},
n = Length[coord];
indexList = Join[Table[i, {i, up}], Table[j, {j, down}]];
S = D[Extract[T, indexList], coord[[var]]];
S = S +
Sum[
-Sum[
Γ[d, down[[b]], var, met, coord]
        Extract[T, ReplacePart[indexList , d, Length[up] + b]],
{d, 1, n}],
{b, Length[down]}
] +
Sum[
Sum[
Γ[up[[b]], d, var, met, coord] Extract[T, ReplacePart[indexList , d, b]],
{d, 1, n}],
{b, Length[up]}
];
If[up == {} && down == {}, S = D[T[[1]], coord[[var]]], S]
]
CovariantD[T_, up_, down_, met_, coord_] :=
 Table[CovariantD[T, up, down, var, met, coord], {var, 1, Length[coord]}]
```

```
(* R[μ_,ν_,ρ_,σ_,met_,coord_] gives
 Subsuperscript[R, abc,    d] for the given metric *)
R[μ_, ν_, ρ_, σ_, met_, coord_] :=
 D[Γ[σ, μ, ρ, met, coord], coord[[ν]]] - D[Γ[σ, ν, ρ, met, coord], coord[[μ]]] +
  Sum[Γ[α, μ, ρ, met, coord] Γ[σ, α, ν, met, coord] -
    Γ[α, ν, ρ, met, coord] Γ[σ, α, μ, met, coord], {α, 1, Length[coord]}]
RiemannTensor[met_, coord_] := Table[R[μ, ν, ρ, σ, met, coord], {μ, 1, Length[coord]},
  {ν, 1, Length[coord]}, {ρ, 1, Length[coord]}, {σ, 1, Length[coord]}]
```

```
(* R[μ_,ν_,met_,coord_] gives the Ricci
 Subsuperscript[tensorR, ab,   ] for the given metric *)
R[μ_, ρ_, met_, coord_] := Sum[R[μ, ν, ρ, ν, met, coord], {ν, 1, Length[coord]}]
RicciTensor[met_, coord_] :=
 Table[R[μ, ρ, met, coord], {μ, 1, Length[coord]}, {ρ, 1, Length[coord]}]
```

```
(* R[met_,coord_] gives the Ricci Scalar R*)
R[met_, coord_] := Module[{imet, n}, n = Length[coord];
  imet = Inverse[met];
Return[Sum[imet[[aa, bb]] R[aa, bb, met, coord], {aa, 1, n}, {bb, 1, n}]]]
```

```
(*This calculates the extrinsic curvature assuming that the surface is defined as being a
k[a_,b_,n_,met_,coord_] := If[a ==1 || b ==1,0,CovariantD[n,{},{b},a,met,coord]]
```

```
(*converts down index object coordinate system*)
ConvertVector[OldVector_,OldCoords_,NewCoords_]:=Table[Sum[D[OldCoords[[i]],a]OldVector
ConvertVector1[OldVector_,OldCoords_,NewCoords_]:=Table[Sum[D[OldCoords[[i]],a]OldVecto
ConvertVector2[OldVector_,OldCoords_,NewCoords_]:=Table[Sum[D[NewCoords[[i]],a]OldVecto
GenerateJacobian[OldCoords_,NewCoords_]:=Table[D[OldCoords[[i]],a],{i,Length[OldCoords]
GenerateJacobian[OldCoords_,NewCoords_]:=Table[D[NewCoords[[i]],a],{i,Length[NewCoords]
Convert2Tensor[OldMetric_,OldCoords_,NewCoords_]:=Table[Sum[D[OldCoords[[i]],a]D[OldCoo
```