## Fichier des régies des sols et cultures

RegieDesSolsEtCultures

La régie des sols et des cultures met en commun les différents apports pour pouvoir les calculer le carbone organique de la zone pour laquelle la régie a été créée.

* calculer\_apport\_annuel\_en\_carbone\_de\_la\_regie(self) :

Prends chacun des types d’apport et calcul la somme des apports et la multiplie par le facteur de travail du sol.

* calculer\_coeffcient\_humification\_residus\_cultures(self) :

Permet d’aller chercher le coefficient d’humification sans connaitre la culture qui est passée à la régie.

* set\_annee\_de\_culture(self, année\_de\_culture) :

Attributs l’année de culture à la régie pour qu’elle soit utilisée pour le bilan.

* generer\_bilan\_regie(self) :

Retourne les apports de chaque type et l’année de culture du bilan

CulturePrincipale

Pour les cultures principales, si l’utilisateur n’a pas entré le pourcentage de la tige ou paille qui a été exportée ou le taux de matière sèche (tiré du pourcentage d’humidité), le programme va chercher l’information par défaut extraite de la table des cultures dans le constructeur.

* calculer\_apport\_en\_carbone\_culture\_principale(self) :

Dans cette méthode pour déterminer la quantité de carbone qui sera apportée au champ, on soustrait à 1 la proportion exportée [0,1] basée sur le pourcentage.

La procédure de calcul de l’apport étant différente pour les plantes fourragères on détermine si elle est ou non une plante fourragère. Le résultat est en kg/m2.

N’est pas une plante fourragère :

Quantité de carbone partie récoltée : conversion du rendement de tonne par hectare en kg par m2 \* taux de carbone de chaque partie (0.45) \* (1-(pourcentage humidité/100)).

Quantité de carbone partie de la tige non-récoltée : Quantité de carbone partie récoltée \* ratio de la partie de la tige non-récoltée / ratio de la partie récoltée \* proportion de la tige laissée au champ.

Quantité de carbone partie racinaire : Quantité de carbone partie \* ratio de la partie racinaire / ratio de la partie récoltée.

Quantité de carbone partie extra racinaire : Quantité de carbone partie \* ratio de la partie extra racinaire / ratio de la partie récoltée.

Si le produit n’est pas récolté :

Quantité de carbone produit + quantité de carbone de de la tige non-récoltée + quantité de la racine + quantité de carbone extra racinaire

Si le produit est récolté :

Quantité de carbone de de la tige non-récoltée + quantité de la racine + quantité de carbone extra racinaire

Est une plante fourragère :

Quantité de carbone partie récoltée : conversion du rendement de tonne par hectare en kg par m2 \* taux de carbone de chaque partie (0.45) \* (1-(pourcentage humidité/100)) \* proportion tige laissée au champ.

Quantité de carbone partie racinaire : Quantité de carbone partie \* ratio de la partie racinaire / ratio de la partie récoltée.

Quantité de carbone partie extra racinaire : Quantité de carbone partie \* ratio de la partie extra racinaire / ratio de la partie récoltée.

Si c’est la dernière année de rotation de plante fourragère :

Quantité de carbone produit + quantité de la racine + quantité de carbone extra racinaire

Si ce n’est pas la dernière année de rotation de plante fourragère:

Quantité de carbone produit + quantité de carbone extraracinaire

* get\_coefficient\_calcul(self) :

Permet d’aller chercher les coefficients de la culture sans besoin de connaitre la culture.

* generer\_bilan\_culture\_principale(self) :

Permet de générer le sommaire du contenu de la régie.

CultureSecondaire

* calculer\_apport\_en\_carbone\_culture\_secondaire(self) :

Calcul l’apport de la culture secondaire s’il y en a une. Pour ce faire, on calcul l’apport comme une plante non fourragère qui est laissée au champ.

* get\_coefficient\_calcul(self) :

Permet d’aller chercher les coefficients relatifs à la culture secondaire sans nécessairement la connaitre.

* est\_non\_null(self) :

Détermine s’il y a ou non une culture secondaire.

* generer\_bilan\_culture\_secondaire(self) :

Permet de générer le sommaire du contenu de la régie.

Amendements

* calculer\_apport\_en\_carbone\_amendements(self) :

Additionne les apports de chaque amendement s’il y en a pour en faire un apport unique pour cette source.

* generer\_bilan\_amendements(self) :

Permet de générer le sommaire du contenu de la régie en listant tous les amendements.

Amendement

* calculer\_apport\_en\_carbone(self) :

Quantité carbone amendement = Apport (t/ha) \* carbone totale (kg/t) \* conversion ha vers m2

* generer\_bilan\_amendement(self) :

Permet de générer le sommaire du contenu de la régie en décrivant le type d’amendement et son apport.

Travail du sol

* calculer\_facteur\_apport\_en\_carbone\_travail\_du\_sol(self) :

Retourne le facteur par lequel multiplier le total des apports selon le type de travail effectué.

## Fichier de l’organisation

EntrepriseAgricole

L’entreprise est composée de champs, du nom de l’entreprise et de sa taille.

* appliquer\_les\_regies\_pour\_la\_duree\_de\_la\_simulation(self, annee\_initiale, annee\_finale) :

Indique aux champs d’appliquer les régies selon l’année initiale et l’année finale qui sont passées en paramètre à cette méthode.

* \_\_calculer\_moyenne\_ponderee\_entreprise\_via\_la\_superficie\_champs(self, bilans\_des\_champs) :

Méthode privé utilitaire qui calcule la moyenne pondérée des gains/pertes des champs de l’entreprise via leur superficie de la façon suivante :

Un bilan annuel moyen pondéré des champs a déjà été calculé en amont, on calcule donc la pondération du champ par rapport à la taille de l’entreprise (bilan[taille\_du\_champ]/self.\_taille\_entreprise). Pour chaque année de bilan annuel, on additionne le bilan annuel de ce champ multiplié par la pondération du champ à l’entrée de la liste du champ qui correspond à l’année du bilan. Si la liste n’a pas encore de donnée pour cette année, on effectue le calcul et on l’insère à la fin de la liste.

* \_\_calculer\_la\_taille\_de\_l\_entreprise(self) :

Méthode privée appelée par le constructeur pour déterminer la taille de l’entreprise. Elle additionne simplement la taille de tous les champs de l’entreprise.

* generer\_le\_bilan\_entreprise(self) :

Permet de générer le sommaire des résultats des bilans sous un format acceptable pour la réponse (JSON) soit le bilan pondéré de l’entreprise qui est aussi calculé dans cette méthode en créant une liste des bilans de champs qui est ensuite passée en paramètre à la méthode privée \_\_calculer\_moyenne\_ponderee\_entreprise\_via\_la\_superficie\_champs.

Champs

Le champ est composé de zones de gestion, du nom et de la taille du champ.

* appliquer\_les\_regies\_pour\_la\_duree\_de\_la\_simulation(self, annee\_initiale, annee\_finale) :

Indique aux zones de gestion d’appliquer les régies selon l’année initiale et l’année finale qui sont passées en paramètre à cette méthode.

* \_\_calculer\_moyenne\_ponderee\_champs\_via\_superficie\_zone\_de\_gestion(self, bilan\_des\_zones) :

Méthode privée utilitaire qui calcule la moyenne pondérée des gains/pertes des zones de gestion des champs via leur superficie de la façon suivante :

Un bilan annuel de carbone a déjà été calculé en amont, on calcule donc la pondération du champs par rapport à la taille de l’entreprise (bilan\_de\_zone[taille\_de\_la\_zone]/self.\_taille\_du\_champs). Pour chaque année de bilan annuel, on additionne le bilan annuel de la zone multiplié par la pondération de la zone à l’entrée de la liste de zone qui correspond à l’année du bilan. Si la liste n’a pas encore de donnée pour cette année, on effectue le calcul et on l’insère à la fin de la liste.

* \_\_calculer\_la\_taille\_du\_champs(self) :

Méthode privée appelée par le constructeur pour déterminer la taille du champ. Elle additionne simplement la taille de toutes les zones du champ.

* get\_taille\_du\_champs(self) :

Retourne la taille du champ calculé dans le constructeur.

* generer\_le\_bilan\_des\_champs(self) :

Permet de générer le sommaire des résultats des bilans sous un format acceptable pour la réponse (JSON) soit le bilan pondéré du champ qui est aussi calculé dans cette méthode en créant une liste des bilans de zones de gestion qui est ensuite passé en paramètre à la méthode privée \_\_calculer\_moyenne\_ponderee\_champs\_via\_superficie\_zone\_de\_gestion. Ce bilan inclut aussi la taille de champ, le nom du champ et son nombre de zone de gestion.

ZoneDeGestion

La zone de gestion est composée de plusieurs attributs donc certain diffèrent selon ce qui est passé en entrée au constructeur. Il y a notamment la constante qui représente le facteur de conversion entre la matière organique et le carbone organique. La masse volumique apparente qui est sélectionnée par défaut si aucune n’est entrée en entrée, même chose pour la profondeur. Les autres entrées sont le taux de matière organique, la classe texturale, la classe de drainage, la taille de la zone, la régie de projection et la régie de réchauffement qui peut être vide. D’autre paramètres comme le coefficient de minéralisation des pools jeunes et vieux sont des attributs générés par le constructeur. La liste qui va contenir les régies pour la durée de la simulation est aussi initialisée dans le constructeur.

* appliquer\_les\_regies\_pour\_la\_duree\_de\_la\_simulation(self, annee\_initiale, année\_finale) :

Ajoute les années de réchauffement dans une liste pour l’initialisation des pools de carbone et assigne les régies selon la rotation donnée pour la période de simulation.

* \_\_calculer\_carbone\_organique\_initial\_du\_sol(self):

facteur\_conversion\_de\_g\_cm2\_a\_kg\_m2 = 10000/1000

conversion\_pourcentage\_taux\_matiere\_organique = 100

facteur\_conversion\_de\_matiere\_organique\_a\_carbone\_organique\_du\_sol = 1 /1.714

carbone initiale du sol = masse volumique apparente \* profondeur \* (taux matière organique / conversion pourcentage taux matière organique) \* facteur conversion de matière organique à carbone organique du sol \* facteur conversion de g/cm2 à kg/m2.

* \_\_calculer\_carbone\_organique\_du\_sol\_pour\_la\_duree\_de\_la\_simulation(self) :

Calcule l’état des pools, documente les apports et l’évolution de la matière organique au fil de la simulation. Les listes qui contiennent les résultats sont initialisées au début de la fonction. Établie d’abord les coefficients d’humification des différents pools de carbone. Établie les pools jeunes initiaux à l’aide de la fonction \_\_calculer\_pools\_carbone\_jeunes\_initiaux(). Additionne tous les pools pour avoir un pool jeune total initial. Calcule le pool de carbone stable à l’aide de la fonction \_\_calculer\_pool\_carbone\_stable\_initial(). Ensuite les apports en carbones sont calculés pour cette année initiale. Redéfinis les apports pour qu’ils respectent les différents pools. Calcule ensuite pour chaque année de la simulation les différents pools jeunes à l’aide de la formule pool\_jeune\_x = (apport+état du pool) \* e^(- coefficient minéralisation pool jeune \* facteur climatique) et le pool stable où on enlève à ce pool le coefficient humification \* (coefficient minéralisation pool jeune / (coefficient minéralisation pool stable – coefficient minéralisation pool jeune)) \* (apport + l’état du pool jeune). Le résultat est ensuite multiplié par e^(-coefficient minéralisation pool stable \* facteur climatique). Par la suite, on additionne à ce résultat le coefficient humification \* (coefficient de minéralisation du pool jeune/ (coefficient minéralisation du pool stable – coefficient minéralisation du pool jeune)) \* (apport + l’état du pool jeune) \* e^(-coefficient minéralisation pool jeune \* facteur climatique). On additionne ensuite les pools jeunes et le pool stable pour avoir le carbone total qui est ensuite transformé en pourcentage de matière organique en multipliant par le facteur de conversion kg/m2 à t/ha en divisant le résultat par 1/1.714 où 1.714 est le facteur de conversion entre la matière organique et le carbone organique du sol.

* \_\_calculer\_pools\_carbone\_jeune\_initiaux(self):

Fonction qui fait la moyenne des années de réchauffement ou d’une rotation s’il n’y a pas de réchauffement pour initialiser les pools jeunes avec la formule état du pool + (apport / coefficient minéralisation pool jeune \* facteur climatique)

* \_\_calculer\_pool\_carbone\_stable\_initial(self, pool\_carbone\_jeune\_initial):

Calcule le pool stable en soustrayant le pool jeune du carbone total initial.

* \_\_calculer\_facteur\_climatique(self):

Calcule le facteur climatique selon la municipalité à l’aide de la formule suivante facteur de la température du sol \* facteur de l’humidité du sol.

* \_\_calculer\_bilan\_annuel\_moyen(self, carbone\_organique\_du\_sol\_pour\_la\_duree\_de\_la\_simulation):

Calcul le bilan annuel moyen en additionnant tous les bilans annuels et divisant le résultat par le nombre de bilan.

* \_\_calculer\_teneur\_apres\_rechauffement(self, matiere\_organique\_du\_sol\_pour\_la\_duree\_de\_la\_simulation) :

Retourne le pourcentage de matière organique du premier bilan.

* \_\_calculer\_teneur\_finale\_projetee(self, matiere\_organique\_du\_sol\_pour\_la\_duree\_de\_la\_simulation):

Retourne le pourcentage de matière organique de la dernière année projetée.

* \_\_calculer\_difference\_entre\_teneur\_initiale\_et\_finale(self, carbone\_organique\_du\_sol\_pour\_la\_duree\_de\_la\_simulation, teneur\_finale\_projetee):

Calcule la différence entre le pourcentage de matière organique du premier et du dernier bilan

* \_\_calculer\_moyenne\_de\_chaque\_annee\_de\_rotation\_dans\_la\_projection(self, carbone\_organique\_de\_sol\_pour\_la\_duree\_de\_la\_simulation):

Pour la durée de la projection, calcule la moyenne pour chaque année de la rotation utilisée dans la projection. Pour ce faire, on additionne dans une liste secondaire les teneurs en COS pour chaque année de la rotation qui se répète au travers de la projection et on divise par le nombre de fois qu’elle s’est répétée.

* \_\_calculer\_moyenne\_des\_apports\_des\_cultures\_principales\_pour\_la\_simulation(self, apports\_de\_la\_culture\_principale\_aerien, apports\_de\_la\_culture\_principale\_racinnaire) :

Calcule la moyenne des apports des cultures principales.

* \_\_calculer\_moyenne\_des\_apports\_des\_cultures\_secondaires \_pour\_la\_simulation(self, apports\_de\_la\_culture\_secondaire\_aerien, apports\_de\_la\_culture\_secondaire\_racinnaire) :

Calcule la moyenne des apports des cultures secondaires.

* \_\_calculer\_moyenne\_des\_apports\_des\_cultures\_principales\_pour\_la\_simulation(self, apports\_des\_amendments) :

Calcule la moyenne des apports des amendements.

* get\_taille\_de\_la\_zone(self):

Permet d’aller chercher la taille de la zone.

* generer\_le\_bilan\_de\_zone(self):

Permet de générer le sommaire des résultats de la zone. Ceci inclus : le bilan de carbone de la zone pour la simulation, le bilan des apports des cultures principales, le bilan des apports des cultures secondaires, le bilan des apports des amendements, le bilan annuel moyen pour la zone, la teneur finale projetée, la teneur initiale projetée, la différence entre la teneur initiale et la teneur finale, la moyenne de chaque année de rotation, la taille de la zone de gestion, le taux de matière organique initial, la classe texturale, la classe de drainage, le bilan des régies de projections et le bilan des régies historiques.

## Fichier décrivant les tables de la base de données

Ce fichier décrit les tables de la base de données afin qu’elles soient gérées par l’ORM de SQLAlchemy. Les classes qui héritent de la base déclarative vont pouvoir être appelé directement pour référé à la table qu’ils décrivent. Leurs attributs pourront aussi être accéder directement en faisant le nom de la classe suivit de l’opérateur « . » et du nom de l’attribut qui décrit la colonne. Chaque colonne de la table est initialisée en assignant la classe Column en passant en paramètre les attributs qui décrivent la colonne (son type, si elle est une clé primaire et dans certain cas une clé étrangère). Il faut aussi préciser le nom que la table prendra dans le système de gestion de base de données. Avec cette explication, le code devrait être relativement simple à comprendre.

## Fichier décrivant la mise en place du lien entre la base de données et l’ORM

Le lien est créé entre la base de données et l’ORM avec la création de l’engine et de la « Session ».

## Fichier de requête vers la base de données

Ce fichier décrit les requêtes effectuées à la base de données pour aller chercher certains attributs qui s’y trouve dont on a besoin pour les calculs d’ICBM. La structure du code dans ce fichier est relativement simple et élégante. Il faut d’abord créer une instance de session. Ensuite, selon les besoins de la requête, on utilise la classe qui correspond à la table que l’on filtre avec « .filter » et « .one » pour aller chercher les données d’une entrée en particulier, ou on va chercher toutes les données d’une colonne de la table avec « .all ».

## Fichier de fonction utilitaire pour la base de données

* add\_column(database\_name, table\_name, column\_name, data\_type):

Ajoute une colonne à la base de données lorsqu’on lui passe le nom de la base de données, le nom de la table, le nom de la nouvelle colonne et son type. Tous les types ne sont pas gérés mais il suffit d’en créer un qui respecte le même format soit

if data\_type == « String qui référera au type »

data\_type\_formatted = « type SQL correspondant »

* change\_column\_name(database\_name, table\_name, column\_name, new\_column\_name):

Cette fonction change le nom de la colonne lorsqu’on lui passe en paramètre le nom de la base de données, le nom de la table, le nom de la colonne dont on souhaite changer le nom et le nouveau nom de la colonne.

* drop\_table(database\_name, table\_name):

Cette fonction sert à supprimer une table si on lui passe en paramètre le nom de la base données et le nom de la table que l’on souhaite supprimer.

* csv\_to\_db\_municipalite():

Cette fonction ne devrait pas être utilisée tel qu’elle mais sert plutôt de modèle pour ajouter des données contenues dans un fichier « .csv » vers une base de données.

## Fichier app qui décrit l’API

Ce fichier sert à établir les routes auxquelles l’API va répondre à mapper les données du POST pour qu’elles puissent être interprétées par le reste de l’API sous forme d’objet Python. L’inverse est aussi vrai pour la réponse au POST.

* post\_simulation() :

Reçois la requête POST JSON et l’envoie vers la fonction qui s’occupe de calculer les résultats souhaités par le format de réponse avant de retourner cette réponse à l’appelant.

* ajout\_amendement() :

Reçois la requête POST JSON et l’envoie vers la fonction qui s’occupe de calculer les résultats souhaités par le format de réponse avant de retourner cette réponse à l’appelant.

* post\_amendement() :

Reçois la requête POST JSON et l’envoie vers la fonction qui s’occupe de calculer les résultats souhaités par le format de réponse avant de retourner cette réponse à l’appelant.

* get\_parametres\_defauts\_culture\_principale() :

Reçois la requête POST JSON et l’envoie vers la fonction qui s’occupe de calculer les résultats, elle formate la réponse selon les résultats obtenues avant de retourner cette réponse à l’appelant.

* get\_municipalite() :

Retourne à l’appelant toutes les municipalités supportées par l’API par leurs noms à l’aide d’un appel à une fonction qui va chercher cette information dans la base de données.

* get\_groupe\_textural() :

Retourne à l’appelant tous les groupes texturaux supportés par l’API par leurs noms à l’aide d’un appel à une fonction qui va chercher cette information dans la base de données.

* get\_classe\_de\_drainage() :

Retourne à l’appelant toutes les classes de drainages supportées par l’API par leurs noms à l’aide d’un appel à une fonction qui va chercher cette information dans la base de données.

* get\_culture\_principale() :

Retourne à l’appelant toutes les cultures principales supportées par l’API par leurs noms à l’aide d’un appel à une fonction qui va chercher cette information dans la base de données.

* get\_travail\_du\_sol() :

Retourne à l’appelant tous les type de travail du sol supportés par l’API par leurs noms à l’aide d’un appel à une fonction qui va chercher cette information dans la base de données.

* get\_culture\_secondaire() :

Retourne à l’appelant toutes les cultures secondaires supportées par l’API par leurs noms à l’aide d’un appel à une fonction qui va chercher cette information dans la base de données.

* get\_amendement() :

Retourne à l’appelant tous les amendements supportés par l’API par leurs noms à l’aide d’un appel à une fonction qui va chercher cette information dans la base de données.

* get\_amendement\_ajoutes() :

Retourne à l’appelant tous les amendements ajoutés à la base de données par l’utilisateur à l’aide d’un appel à une fonction qui va chercher cette information dans la base de données.

* is\_decimal\_number(string) :

Permet de vérifier si un string peut être transformé de façon sécuritaire en nombre à virgule flotante.

* \_\_launch\_icbm\_simulation(data):

Cette fonction établie un objet GestionSimulation qui permet de lancer les calculs et retourne le résultat de ceux-ci. Les données sont d’abord mappées en objet Python.

* \_\_ajout\_amendement(data) :

Vérifie que l’amendement ajouté correspond à un amendement valide.

* \_\_ajout\_amendements(data) :

Vérifie que les amendements ajoutés correspondent à des amendements valides.

* \_\_simulation\_mapping(data):

Les données reçues en paramètre par cette fonction sont transformées en objet Simulation si les données respectent les types désirés.

* \_\_champs\_mapping(data) :

Les données reçues en paramètre par cette fonction sont transformées en objet Champs si les données respectent les types désirés.

* \_\_zone\_de\_gestion\_mapping(data) :

Les données reçues en paramètre par cette fonction sont transformées en objet ZoneDeGestion si les données respectent les types désirés.

* \_\_regie\_sol\_et\_culture\_mapping(data, municipalite):

Cette fonction permet de gérer le mapping de chaque sous partie du calcul. La culture principale ayant besoin d’information supplémentaire sur la nature de la plante et où se situe l’année dans une l’ensemble des régies pour bien être mappé, cette information est ajoutée au mapping de la culture pour obtenir une information valide lors des calculs.

* \_\_travail\_du\_sol\_mapping(data) :

Les données reçues en paramètre par cette fonction sont transformées en objet TravailDuSol si les données respectent les types désirés.

* \_\_amendements\_mapping(data) :

Les données reçues en paramètre par cette fonction sont transformées en objet Amendement si les données respectent les types désirés. Les amendements sont optionnels, il est donc possible de ne pas en avoir. Dans ce cas, on donne None aux attributs de l’amendement et c’est l’objet qui s’occupera de gérer ce cas.

* \_\_culture\_principale\_mapping(data, est\_derniere\_annee\_rotation\_culture\_fourragere, municipalite) :

Les données reçues en paramètre par cette fonction sont transformées en objet CulturePrincipale si les données respectent les types désirés. Dans cette fonction, on attribue un rendement si le rendement est None et prend en considération la culture et la municipalité pour aller chercher le rendement par défaut qui correspond.

* \_\_get\_rendement\_defaut\_municipalite(culture\_principale, municipalite):

Va chercher le bon rendement dans la base de données selon la culture et la municipalité.

* \_\_culture\_secondaire\_mapping(data) :

Les données reçues en paramètre par cette fonction sont transformées en objet CultureSecondaire si les données respectent les types désirés. Il est possible qu’il n’y ait pas de culture secondaire, le nom de la culture est alors None et son rendement de 0.

* If \_\_name\_\_ == \_\_main\_\_ :

La fonction « main » de l’application permet de faire servir l’API en local sur le port 5000.

## Fichier de gestion des simulations

Ce fichier contient la classe Simulation et GestionSimulation. La Simulation contient une entreprise agricole pour laquelle elle applique les régies selon l’année initiale et l’année finale. Elle permet aussi de demander le bilan de l’entreprise qu’elle simule. La classe GestionSimulation contient une liste de simulations et lorsque sa fonction generer\_les\_bilans\_pour\_les\_simulations\_de\_l\_entreprise\_agricole(self) : est appelée cela lance les calculs sur chaque simulation et retourne une liste des bilans de chacune.

## Fichier fonction utilitaire

Ce fichier contient uniquement la fonction is\_decimal\_number qui permet d’identifier les Strings qui respecte le type float et qui ne causeront pas d’erreurs lors du changement de type.

## Fichier de l’interface graphique

Description des variables globales :

* furthest\_left\_tab\_index : Il y a trois variables globales qui portent ce nom, une pour chaque niveau de tab. Cette variable décrit l’index du tab le plus à gauche qui est affiché dans son tab respectif.
* furthest\_right\_tab\_index : Il y a trois variables globales qui portent ce nom, une pour chaque niveau de tab. Cette variable décrit l’index du le plus à droite qui est affiché dans son tab respectif.
* min\_index : Il y a trois variables globales qui portent ce nom, une pour chaque niveau de tab. Cette variable décrit l’index minimum du niveau de tab qui est de 0 dans tous les cas car en Python l’index d’une liste commence à 0.
* max\_index : Il y a trois variables globales qui portent ce nom, une pour chaque niveau de tab. Cette variable décrit l’index maximum du niveau de tab auquel elle réfère. Le maximum réfère à l’index le plus à droite dans la « liste de tab » du Notebook même si celui-ci n’est pas nécessairement affiché.
* nombre\_simulations : Cette variable réfère au nombre de simulation que l’utilisateur a créé. Ce nombre varie lorsque des simulations sont ajoutées ou enlevées.
* duree\_simulation : Cette variable contient une liste de durée qui correspondent à la durée de la simulation respectant la même indexation que les simulations dans le Notebook qui gère ces tabs.
* simulation\_chargees : Booléen qui détermine si les simulations ont été chargé à partir d’un fichier de sauvegarde ou non.
* filename : Nom du fichier de sauvegarde contenant les données à propos de l’entreprise.
* plan\_gestion\_filename : Nom du fichier contenant les données complètes d’un plan de gestion et qui permet de le charger pour pouvoir continuer de travailler dessus sans avoir à tout rentrer de nouveau.
* municipalites\_supportees : Liste de toutes les municipalités acquise via l’API puis triée en ordre alphabétique.
* classes\_texturales\_supportees : Liste de toutes les classes texturales acquise via l’API puis triée en ordre alphabétique.
* classes\_de\_drainage\_supportees : Liste de toutes les classes de drainages acquise via l’API puis triée en ordre alphabétique.
* cultures\_principales\_supportees : Liste de toutes les cultures principales acquise via l’API puis triée en ordre alphabétique.
* types\_travail\_du\_sol\_supportes : Liste de tous les types de travail du sol acquise via l’API puis triée en ordre alphabétique.
* cultures\_secondaires\_supportees : Liste de toutes les cultures secondaires acquise via l’API puis triée en ordre alphabétique.
* amendements\_supportes : Liste de tous les amendements acquise via l’API puis triée en ordre alphabétique.
* nom\_entreprise : Variable qui décrit le nom de l’entreprise.
* nombre\_de\_champs : Variable qui indique le nombre de champs que possède l’entreprise.
* information\_champs : Variable qui contient une liste de dictionnaire qui décrivent le nom de chaque champs, leurs nombre de zones de gestion et l’information sur cette zone de gestion.
* nombre\_de\_champs\_modifie : Booléen qui indique si un champs a bel et bien été ajouté.
* information\_champs\_modifie : Booléen qui indique si un champs a bel et bien été ajouté.

Description des fonctions :

* sp = subprocess.Popen("cd ../ICBM/VirtualEnvironmentSetUp & py \"app.py\"", shell=True)

Démarre le sous-processus qui fait rouler l’API.

* initialize\_globals() :

Définit les valeurs de certaines valeurs globales via l’API puis lance l’interface à l’aide de la fonction initialize\_run\_gui().

* initialize\_run\_gui() :

Définit la fenêtre principale de l’interface et initialise les variables nécessaires à son bon fonctionnement.

* kill(proc\_id) :

Fonction permettant d’arrêter le sous-processus qui fait rouler l’API.

* on\_mousewheel(event) :

Fonction qui sera exécuté lorsque la roue de la souris sera utilisée par l’utilisateur, les canvas et les scrollbars réagiront pour changer ce qui est affiché.

* callback(event) :

Fonction qui sert à gérer l’interaction avec un lien créé à partir d’un label.

* filter\_combobox\_values(combobox, values) :

Fonction qui filtre les entrées de la combobox et retourne les options filtrées dans les valeurs de la combobox si l’entrée n’est pas vide.

* closing\_root\_protocol() :

Arrête l’API et ferme la fenêtre principale de l’interface.

* run\_gui(frame) :

Démarre le fil d’exécution de l’interface.

* show\_entreprise\_set\_up(frame\_entreprise) :

Affiche l’interface qui permet d’entrer les données de l’entreprise.

* get\_information\_entreprise() :

Va chercher les données de l’interface où les données de l’entreprise sont entrées et vérifie s’il y a des erreurs de types.

* show\_creation\_champs(frame\_champs\_list) :

Affiche l’interface qui permet d’entrer les données de champs selon ce qui a été entrer dans l’interface d’entreprise.

* get\_information\_champs(scrollable\_frame) :

Va chercher les données de l’interface où les données de champs sont entrées et vérifie s’il y a des erreurs de types.

* show\_creation\_zone\_de\_gestion(zone\_de\_gestion\_mainframe, fill\_fields=False) :

Affiche l’interface qui permet d’entrer les données des zones de gestion selon ce qui a été entré dans l’interface champ. Si l’utilisateur annule la sauvegarde du fichier entreprise qui est l’étape suivante celle-ci les champs sont remplis de sorte que l’utilisateur n’a pas besoin de remplir l’information de nouveau.

* get\_information\_zone\_de\_gestion(scrollable\_frame) :

Va chercher les données de l’interface où les données de l’entreprise sont entrées et vérifie s’il y a des erreurs de types et que les données entrées sont supportées par l’API.

* zone\_frame\_generation(zone\_gestion\_frame) :

Génère les frames des zones avec tous les champs qu’elles contiennent et les remplissent si nécessaire.

* show\_creation\_des\_regies(parent\_frame\_tabs, simulations=None) :

Affiche l’interface qui permet d’entrer les données à propos des régies. Cette interface s’adapte à ce qui a été entré dans les étapes précédentes.

* add\_new\_simulation\_tab(event) :

Advenant un clic gauche sur le dernier tab qui est identifié avec un « + » par l’utilisateur, cette fonction va ajouter un nouveau tab où il est possible de modifier les données de régies. Entre le clic et l’affichage du tab il y a une interface qui permet d’identifier certains paramètres qui est décrite dans la fonction duree\_de\_la\_simulation().

* duree\_de\_la\_simulation() :

Fonction qui affiche l’interface qui permet de définir, le nom de la simulation, l’année initiale de projection, la durée de la projection, le numéro de la simulation à copier.

* readd\_simulation() :

Permet de remettre les tabs et la variable globale nombre de simulation à l’état précédent l’appel à la fonction add\_new\_simulation\_tab(event).

* get\_duree\_de\_la\_simulation() :

Va chercher les entrées de l’interface avec les paramètres de durée et vérifie s’il y a des erreurs de types. Si c’est le cas, on ramène la simulation à l’état précédant l’ajout à l’aide de la fonction readd\_simulation.

* delete\_simulation\_tab(event) :

Cette fonction supprime le tab de la simulation sous le curseur lorsque l’utilisateur fait un clic droit. Seul le dernier tab (celui avec un « + ») ne peut pas être supprimé. La fonction va ajuster certaines variables globales en fonction du nombre de tab présentement actif.

* set\_up\_simulation(simulation\_notebook, simulation\_copie, simulation\_en\_tete\_de\_liste) :

Fonction qui met en place les éléments relatifs à l’affichage de la simulation et qui ajoute les champs au Notebook de champs s’il y en avait. Elle gère aussi les champs et simulations qui sont affichés ainsi que l’activation des boutons de scroll.

* add\_new\_champs\_tab(event) :

Fonction qui définit l’ajout d’un tab lorsque le dernier tab du Notebook de champs est cliqué par l’utilisateur. La mise en place des éléments se fait cependant dans la fonction set\_up\_new\_champs().

* delete\_champs\_tab(event) :

Cette fonction supprime le tab champ en dessous du curseur lors d’un clic droit de l’utilisateur et son équivalent dans chaque simulation. Il supprime les données de ce champ dans la variable globale information\_champs et décrémente la variable globale nombre\_de\_champs. Il y a aussi une gestion des tabs affichés et des boutons de scroll pour assurer le bon fonctionnement du système de tab. La fonction enlève aussi le champ de la section réchauffement de l’interface.

* set\_up\_new\_champs() :

Cette fonction s’occupe de fournir les interfaces nécessaires pour entrer les données du nouveau champ soit son nom, son nombre de zone de gestion, et (à l’aide d’une autre interface) les informations sur les caractéristiques de la zone. Elle vérifie que les informations entrées sont valides avant de procéder à la création du nouveau champ au travers des simulations.

* revenir\_a\_etat\_initial() :

Cette fonction ramène les variables globales à leur état avant l’ajout du tab lorsque que l’utilisateur ferme les interfaces de création du nouveau champ.

* creation\_des\_zones\_de\_gestion\_du\_nouveau\_champs() :

Fonction qui s’occupe de créer l’interface spécifique de la zone du nouveau champ selon les informations entrées dans la première interface de création de champ.

* zone\_frame\_generation() :

Fonction qui s’occupe d’ajouter les champs nécessaires pour la création de nouveaux champs dans l’interface de création de zone.

* add\_new\_zone\_tab() :

Fonction qui vérifie si les données entrées par l’utilisateur dans la zone de gestion sont valides et qui ajuste l’affichage des tabs de champs et ajoute les nouveaux tabs dans les Notebooks appropriés. Les modifications sont sauvegardées au fichier d’entreprise.

* set\_up\_champs(zone\_notebook, nombre\_de\_zone, champs\_notebook, simulation\_copie=None, index\_champs=None) :

Cette fonction affiche les éléments qui relève de la zone de gestion dans l’interface qui décrit les régies en fonction de ses paramètres d’entrée. Pour une copie de simulation, on extrait dans cette fonction les données de copie pour pouvoir les mettre dans les champs appropriés plus dans la fonction set\_up\_regies\_projections().

* add\_new\_zone\_de\_gestion\_tab(event) :

Fonction qui répond au clic gauche d’un utilisateur sur le tab « + » des tabs de zones de gestion soit le dernier tab du Notebook et modifie le nombre de zone de gestion du champ parent. La gestion de l’affichage s’effectue dans la fonction set\_up\_new\_zone\_de\_gestion(champs\_index).

* delete\_zone\_de\_gestion\_tab(event) :

Cette fonction supprime le tab sous le curseur lors d’un clic droit de l’utilisateur alors que le curseur est au-dessus d’un tab de cette section et son équivalent dans chaque simulation. Les variables globales sont mises à jour et les modifications sont sauvegardées. Elle gère aussi l’affichage des tabs après la suppression et les boutons de scroll.

* set\_up\_new\_zone de\_gestion(champs\_index) :

Fonction qui présente une interface pour remplir les caractéristiques de la zone de gestion.

* remise\_a\_etat\_initial() :

Fonction qui modifie la variable globale information\_champs pour la ramener à un état valide lorsque l’action d’ajout de zone est annulée.

* add\_new\_tab() :

Cette fonction vérifie les données entrées à propos de la zone de gestion. Si elles sont valides la variable globale information\_champs est modifiée pour refléter l’ajout de la zone. Les données sont sauvegardées. L’affichage du tab et l’ajout dans le réchauffement est aussi effectué par cette fonction.

* set\_up\_regies\_projections(zone\_tab, champs=None, zone\_index=None) :

Cette fonction mets en place le canvas où les régies seront ajoutées ainsi que les boutons pour créer le rapport et éditer les caractéristiques de l’entreprise.

* ajouter\_une\_annee\_a\_la\_rotation(scrollable\_frame\_projection, zone=None, zone\_rechauffement=None) :

Cette fonction ajoute une année à la rotation en appelant la fonction add\_regies\_projection(). Initialement, le bouton « enlever une année de rotation » qu’elle ajoute ne peut pas être utilisé car il n’y a qu’une seule rotation mais celui-ci se débloque à l’ajout d’une année de rotation. Cette fonction est utilisée autant par la projection que le réchauffement.

* enlever\_une\_annee\_a\_la\_rotation(scrollable\_frame\_projection) :

Fonction qui permet d’enlever une année à la rotation en tenant compte de si elle est ou non la dernière rotation restante.

* appliquer\_les\_regies\_aux\_autres\_zones\_rechauffement(champ\_label\_frame, zone\_courante\_index):

Cette fonction applique la rotation du champ qui correspond au bouton après avoir vérifié que toutes les entrées sont valides.

* set\_up\_regies\_rechauffement(réchauffement\_frame, show\_regie\_historique, simulations) :

Cette fonction affiche la partie de l’interface qui permet à l’utilisateur de saisir les données de réchauffement. C’est-à-dire, les données qui servent à initialiser le modèle ICBM pour que les calculs soient représentatifs de la réalité. La fonction affiche pour toutes les zones des champs des simulations à l’aide de la fonction ajouter\_une\_annee\_a\_la\_rotation(rechauffement\_champs\_label\_frame, zone\_rechauffement=zone\_rechauffement) qui va permettre l’ajout dynamique d’année de rotation.

* ajouter\_des\_amendements(amendement\_frame, amendements=None) :

Fonction qui met en place un amendement initial à la régie.

* on\_amendment\_selected(event) :

Fonction qui insère un apport par défaut de 10 t/ha lorsqu’un amendement est sélectionné dans la combobox.

* ajouter\_amendement\_regie(amendement\_frame) :

Fonction qui permet l’ajout dynamique de plus d’amendement après l’amendement initial.

* on\_amendment\_selected(event) :

Fonction qui insère un apport par défaut de 10 t/ha lorsqu’un amendement est sélectionné dans la combobox.

* add\_regies\_projection(scrollable\_frame, index, regie=None) :

Cette fonction affiche les champs nécessaires pour que l’utilisateur entre les données des régies de projection et réchauffement. L’ajout est gérer par la fonction ajouter\_une\_annee\_a\_la\_rotation(scrollable\_frame\_projection, zone\_rechauffement= None).

* on\_culture\_principale\_selected(event) :

Fonction qui insère un rendement, un pourcentage de tige exporté, si oui ou non la production a été récoltée, un pourcentage d’humidité et un travail du sol par défaut lorsque la culture principale est choisie via la combobox.

* on\_culture\_secondaire\_selected(event) :

Fonction qui ajoute un rendement par défaut de 1 t/ha lorsqu’une culture secondaire est sélectionnée via la combobox.

* appliquer\_les\_regies\_aux\_autres\_zones(zone\_tab):

Fonction qui permet à l’utilisateur d’insérer régies d’une zone de gestion à toutes les zones de gestion d’un champ dans la section régies projetées.

* get\_information\_toutes\_les\_simulations() :

Fonction qui va chercher toutes les informations des simulations formattées par la fonction get\_information\_simulation, s’il n’y a pas d’entrées invalides elle envoie une requête POST à l’API avec ces données et reçoit la réponse.

* sauvegarder\_toutes\_les\_simulation() :

Fonction qui va chercher toutes les informations des simulations formattées par la fonction get\_information\_simulation, s’il n’y a pas d’entrées invalides elle retourne les données sous un format JSON pour la sauvegarde.

* get\_information\_simulation(simulation\_frame, index\_simulation, simulation\_unique) :

Fonction qui formatte les résultats qui ont été obtenus par les fonctions get\_regies et get\_regies\_rechauffement, ainsi que les données de certaines variables globales pour former un dictionnaire qui synthétise les données de la simulation sous un format qui respecte l’API. Elle retourne un tuple de la simulation et une liste des erreurs de validités.

* get\_regies (regies\_frame, champs\_index, zone\_index, simulation\_index) :

Cette fonction va chercher les données d’entrée des régies de projection et les retourne formatter sous la forme d’un dictionnaire ainsi qu’une liste des erreurs de validités dans un tuple.

* scroll\_right\_simulation() :

Fonction qui gère la navigation des tabs de la simulation vers la droite.

* scroll\_left\_simulation() :

Fonction qui gère la navigation des tabs de la simulation vers la gauche.

* scroll\_right\_champs() :

Fonction qui gère la navigation des tabs du champ vers la droite.

* scroll\_left\_champs() :

Fonction qui gère la navigation des tabs du champ vers la gauche.

* scroll\_right\_zone() :

Fonction qui gère la navigation des tabs de la zone vers la droite.

* scroll\_left\_zone() :

Fonction qui gère la navigation des tabs de la zone vers la gauche.

* zone\_tab\_management() :

Fonction qui gère les tabs des zones de gestion lorsqu’un tab est changé, tous les tabs sont cachés puis les 4 premières (s’il en a 4 ou plus) sont affichées. Les boutons de scroll sont ajustés en fonction du nombre de tabs affichés.

* get\_regies\_rechauffement

Cette fonction va chercher les données d’entrée des régies de réchauffement et les retourne formattées sous la forme d’un dictionnaire ainsi qu’une liste des erreurs de validités dans un tuple.

* editer\_caracteristique\_physique\_entreprise() :

Cette fonction affiche une interface qui permet d’éditer les caractéristiques de l’entreprise. Les caractéristiques présentes sont chargées dans les champs pour permettre à l’utilisateur de valider s’il désire les changer ou non.

* fenetre\_edition\_ferme() :

Fonction qui permet de fermer la fenêtre d’édition.

* effectuer\_la\_sauvegarde() :

Cette fonction permet de valider les nouvelles entrées avant d’effectuer les modifications à la fenêtre principale et la sauvegarde qui a lieu après la modification des données.

* effectuer\_les\_modifications\_a\_la\_fenetre\_principale() :

Ajuste les noms des champs dans la fenêtre principale selon les nouvelles données entrées dans la fenêtre de modification des caractéristiques.

* menu\_initial\_ogemos(menu) :

Fonction qui affiche le menu principal d’OGEMOS et ses trois différentes options à l’aide de boutons.

* creation\_nouvelle\_entreprise() :

Fonction qui envoie vers la fonction show\_entreprise\_set\_up qui permet de créer une nouvelle entreprise.

* charger\_entreprise() :

Fonction qui va chercher un fichier d’entreprise et charge les données qu’il contient si celui-ci est valide en passant par la fonction question\_ajoute\_regie\_historique.

* charger\_plan\_de\_gestion\_de\_carbone() :

Fonction qui va chercher un fichier de plan de gestion et charge les données qu’il contient si celui-ci est valide en passant par la fonction show\_creation\_des\_regies.

* ajouter\_un\_nouvel\_amendement(data=None) :

Fonction qui affiche une interface qui permet à l’utilisateur d’ajouter un amendement personnalisé à la base de données.

* fenetre\_nouvel\_amendement\_ferme() :

Fonction qui permet de fermer la fenêtre de nouvel amendement.

* get\_nouvel\_amendement() :

Fonction qui va chercher les données du nouvel amendement et vérifie leurs validités avant de les envoyer à l’API. Une fois l’ajout effectué, l’API envoie une réponse sur le succès ou l’échec de l’ajout. Une messagebox informe donc l’utilisateur du succès de l’opération.

* menu\_transfert\_version() :

Fonction qui définit un nouveau menu pour le transfert de version. L’objectif de ce menu est de faciliter le transfert des amendements personnalisés.

* fenetre\_menu\_transfert\_ferme() :

Fonction qui permet de fermer la fenêtre du menu de transfert.

* sauvegarder\_amendement\_dans\_un\_fichier() :

Fonction qui fait un appel à l’API pour obtenir les amendements personnalisés de l’utilisateur et les sauvegardes sous un format JSON.

* charger\_amendement\_a\_partir\_de\_fichier() :

Fonction qui va chercher les amendements dans un fichier qui contient un JSON. Vérifie que le JSON est valide puis l’envoie vers la base de données grâce à un appel à l’API.

* sauvegarder\_attributs\_entreprise\_apres\_creation() :

Cette fonction sauvegarde le fichier avec l’extension « .json » selon le nom entré par l’utilisateur dans l’interface de navigation de système de fichier. Elle sauvegarde les données sur l’entreprise après sa création.

* sauvegarder\_attributs\_entreprise\_apres\_modification() :

Cette fonction sert à sauvegarder automatiquement après une modification aux caractéristiques de l’entreprise à l’aide du fichier créé à la création ou celui chargé lors du chargement. Si aucun fichier d’entreprise n’a été créé ou chargé, une messagebox alerte l’utilisateur d’en créer un sinon la modification ne sera pas sauvegardée.

* sauvegarder\_plan\_de\_gestion\_de\_carbone(simulations) :

Cette fonction sert à sauvegarder le plan de gestion dans un fichier avec l’extension « .json » selon le nom entré par l’utilisateur dans l’interface de navigation de système de fichier.

* retour\_au\_menu\_principal() :

Cette fonction sert à retourner au menu principal tout en réinitialisant les variables globales.

* creation\_du\_rapport(bilan\_reponse) :

Cette fonction crée le bilan sous la forme d’un fichier Excel à partir des données obtenues à partir de l’API sur l’évolution du taux de matière organique, l’état des différents pools de carbone et des apports au fil de la simulation.

* sauvegarde\_rapports\_des\_resultats() :

Cette fonction sert à sauvegarder le rapport selon le nom entré dans l’explorateur de fichier. Indique si l’opération a été un succès ou un échec selon si le fichier était ouvert ailleurs ou non.

# Requête pour modifier la base de données

Pour modifier la base de données, il est possible d’utiliser SQLAlchemy via un interpréteur Python par exemple. Il faut changer l’interpréteur pour que le chemin soit dans le dossier où est le fichier de base de données. Pour ce faire, il faut importer create\_engine et sessionmaker tel que démontré dans la figure suivante :



Figure 1.1 Import create\_engine et sessionmaker

L’étape suivante est de créer l’engine et d’initialiser le sessionmaker. Create\_engine prend en entrée un string qui devrait être comme le suivant « sqlite:///coefficient.db ». La figure suivante montre l’initialisation :



Figure 1.2 Initialisation engine et sessionmaker

Il faut ensuite créer la session comme dans la figure suivante :



Figure 1.3 Création de l’objet session

Il est aussi important d’aussi initialiser la declarative\_base qui représente le modèle sur lequel est basé la base de données tel qu’illustrée dans la figure 1.4 :

![Text

Description automatically generated](data:image/jpeg;base64,/9j/4AAQSkZJRgABAQEAYABgAAD/4RD0RXhpZgAATU0AKgAAAAgABAE7AAIAAAAOAAAISodpAAQAAAABAAAIWJydAAEAAAAcAAAQ0OocAAcAAAgMAAAAPgAAAAAc6gAAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAFNhbXVlbCBDb21lYXUAAAWQAwACAAAAFAAAEKaQBAACAAAAFAAAELqSkQACAAAAAzA5AACSkgACAAAAAzA5AADqHAAHAAAIDAAACJoAAAAAHOoAAAAIAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAyMDIwOjExOjE4IDE0OjUyOjQzADIwMjA6MTE6MTggMTQ6NTI6NDMAAABTAGEAbQB1AGUAbAAgAEMAbwBtAGUAYQB1AAAA/+ELIGh0dHA6Ly9ucy5hZG9iZS5jb20veGFwLzEuMC8APD94cGFja2V0IGJlZ2luPSfvu78nIGlkPSdXNU0wTXBDZWhpSHpyZVN6TlRjemtjOWQnPz4NCjx4OnhtcG1ldGEgeG1sbnM6eD0iYWRvYmU6bnM6bWV0YS8iPjxyZGY6UkRGIHhtbG5zOnJkZj0iaHR0cDovL3d3dy53My5vcmcvMTk5OS8wMi8yMi1yZGYtc3ludGF4LW5zIyI+PHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9InV1aWQ6ZmFmNWJkZDUtYmEzZC0xMWRhLWFkMzEtZDMzZDc1MTgyZjFiIiB4bWxuczpkYz0iaHR0cDovL3B1cmwub3JnL2RjL2VsZW1lbnRzLzEuMS8iLz48cmRmOkRlc2NyaXB0aW9uIHJkZjphYm91dD0idXVpZDpmYWY1YmRkNS1iYTNkLTExZGEtYWQzMS1kMzNkNzUxODJmMWIiIHhtbG5zOnhtcD0iaHR0cDovL25zLmFkb2JlLmNvbS94YXAvMS4wLyI+PHhtcDpDcmVhdGVEYXRlPjIwMjAtMTEtMThUMTQ6NTI6NDMuMDg4PC94bXA6Q3JlYXRlRGF0ZT48L3JkZjpEZXNjcmlwdGlvbj48cmRmOkRlc2NyaXB0aW9uIHJkZjphYm91dD0idXVpZDpmYWY1YmRkNS1iYTNkLTExZGEtYWQzMS1kMzNkNzUxODJmMWIiIHhtbG5zOmRjPSJodHRwOi8vcHVybC5vcmcvZGMvZWxlbWVudHMvMS4xLyI+PGRjOmNyZWF0b3I+PHJkZjpTZXEgeG1sbnM6cmRmPSJodHRwOi8vd3d3LnczLm9yZy8xOTk5LzAyLzIyLXJkZi1zeW50YXgtbnMjIj48cmRmOmxpPlNhbXVlbCBDb21lYXU8L3JkZjpsaT48L3JkZjpTZXE+DQoJCQk8L2RjOmNyZWF0b3I+PC9yZGY6RGVzY3JpcHRpb24+PC9yZGY6UkRGPjwveDp4bXBtZXRhPg0KICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICA8P3hwYWNrZXQgZW5kPSd3Jz8+/9sAQwAHBQUGBQQHBgUGCAcHCAoRCwoJCQoVDxAMERgVGhkYFRgXGx4nIRsdJR0XGCIuIiUoKSssKxogLzMvKjInKisq/9sAQwEHCAgKCQoUCwsUKhwYHCoqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioq/8AAEQgC5ALBAwEiAAIRAQMRAf/EAB8AAAEFAQEBAQEBAAAAAAAAAAABAgMEBQYHCAkKC//EALUQAAIBAwMCBAMFBQQEAAABfQECAwAEEQUSITFBBhNRYQcicRQygZGhCCNCscEVUtHwJDNicoIJChYXGBkaJSYnKCkqNDU2Nzg5OkNERUZHSElKU1RVVldYWVpjZGVmZ2hpanN0dXZ3eHl6g4SFhoeIiYqSk5SVlpeYmZqio6Slpqeoqaqys7S1tre4ubrCw8TFxsfIycrS09TV1tfY2drh4uPk5ebn6Onq8fLz9PX29/j5+v/EAB8BAAMBAQEBAQEBAQEAAAAAAAABAgMEBQYHCAkKC//EALURAAIBAgQEAwQHBQQEAAECdwABAgMRBAUhMQYSQVEHYXETIjKBCBRCkaGxwQkjM1LwFWJy0QoWJDThJfEXGBkaJicoKSo1Njc4OTpDREVGR0hJSlNUVVZXWFlaY2RlZmdoaWpzdHV2d3h5eoKDhIWGh4iJipKTlJWWl5iZmqKjpKWmp6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2uLj5OXm5+jp6vLz9PX29/j5+v/aAAwDAQACEQMRAD8A81ktLS20GOfzLWeaZnAJ80MuNvC8AZGTnPHpmoZNGlSUQpcW8twWVfIRju+bp1AH68d6qSXLyWkNuwXZCzMpA5O7Gc/lT2v5zqAvVKpMrBgVHAI+v0rBRqLr3/4Bu5QfTt/wSymivL80F3bSxANulVmCptGTnKg9OeBzUWo6XLpjKs0kUm4sv7sk4Ixkcgeop51iRUZILa3gjZXDJGGwSwwTySc46c49qivtRm1BgZlRcOz/ACAjlsZ7+1KPteZX2HL2XI7bkmmaRLqiXMiTRwRWyB5ZJFdgoz6IrH8cYHrVjTvD0uo+XtvbWATzmC3Mxf8AfsMZ2hVJA5HJx1+tVdM1EaZcCdbSGeVSGjaR5FMZHcFHX9a19H8TmPV7aTU4rZ4hem5MzRsWgLEbyoU45x0IOO2DXUrHK72ZSggtpfDF9I9rGLm1miCzqz7mDFsgjdt7DsKG8N3g0A6ujK9uqhmHlSrgFtv3mQIeTj5WNMt9ZW20yexGnWkyTsC8shl3kjO08OBxk9vrmnzeI55tPktTaWqtLBHbyTqrb2RCpX+LA+6OgGaWlin8XkP1CziubXRm0+ySGe9RlMcTMQ7iQqPvMcZ474p1jotudXFpJe2l2TDPuETSKInWNiCzMqjGR1BI49Kim8QO8NmlvYWlq9iwaCWIyFkO7d/E5B555FRya1m7Nzb6faW0rJKjmLzMP5ilScM5AwCcYwKen5i1LMPhS8ubyOG0mhuI5Lf7Ss8KSOuzdt+6E353DGNvv05qYeC9Q8+WKae1gMZx++Z13fuzJ025Hygn5gMYwcGqMOvXEUcUTwQTQJbG2aGQNtkTeX5wQchjkEEdPrl0HiCa0WRLS0toYndn8td5Clo2jOCWJ6MTyevtxQ7dA1Hy+GLyO6hhWa3lMtx9nDozbVbAbJyBgFWzn2PpWO6hJGUMHAJAZc4PuM10Ca4E03ULiSaJru+VYkt40b9zhSpck8ZKEqME/ePTFY6XmzTZbT7Nbt5jq/nsmZFxngN2BzyKTt0BX6mnL4S1GHR/7RlAWLy1lYGOQbUbGDu27D1HAYn24OIrjw9LZhZLu8tYYJGUQzEuVnUjO9MKSVHGeBgnHXIqG51f7ZbotzY2sk6IqfasyCRlXAAOG2ngAZ25x780t7rk9/am3ngg8tXDQBVI+zgDG1OfukAZBzyM9cmnpfQFfqSeIdJttI1SW3tL6O5VHK7AH3p/vEqB+Waqafpz6g0uJYreKBPMlmmJ2oMgDoCTkkDAB61Lf6sdRuI7iaztlnBBkkXf++Ix94FiO38IHWnx628d9eT/AGGzMN4NstoEZYsZBAAVgRggHg0tLhrY1bjwlLearNFpW37PBBblpFjmkDM8QbICoz8nceQAPbgVmt4fnga4/tG6trFYJzblpi7B5ByQAiseBjnGORT5PEk9xNObyys7iKcR7rdldUUxrtVl2sCDjI4OOenSoINY8q2ktZdPtLi2aUzJDL5gETEYO0q4bGABgk9BTduga2JLeG2m8L3srWsYuLaWILcKz7mDFsgjdt7DsKVvDd4NAOroyvbqoZh5Uq4Bbb95kCHk4+VjUdrrK22lTWH9m2kqTkF5HMu8kZ2nhwOMnt9c1JN4jnm0+S1Npaq0sEdvJOqtvZEKlf4sD7o6AZodg1uNvdBNixjk1GyeZZFR4Y2cum7ofu4I9duSM9Kvf8IhNa6tp1vqdykMF5LsEhimjJIIBUB4w2fmGDjb79ayjq1wdcXVQsYuFlWUDb8u4YxwfpU91r80/wBlEFtbWYtJmniEAb77bSSSzEnlRTXLfUWpPbaTZr4ogsje2l3CZgpUmZFf5seWWEeQT6gY96pPpcskd3PAEK29wImhUlmG4naRxyMjHrnHHNTLraR6wmoppVksiNvEYMuzfuzv/wBZnOe2ce1EGvy2uqS31raW0JlGGiAdkzkMG+ZichgD1xkdMcVKtZXG93YuWngvU7ySdYDG/kzGDcqSuryDGRlUIXGQMttHvwa59lKOVYYZTgitC31cx2X2W6s7e+iEhkT7QXBjYgBiCjKecDg5HFZ1DGjQNmh0iCSPymkkn2Fw7ZXgfKVIx75BPWp5PDlzGVH2i2ZmkEeFZuDv2Ht2b/62arJqmyxS2Wzt8I/mB8vu3dM/ex29MVZg12V72E3IjSLzgzsqngeb5h7+v6VyS9svhOmPsmve3sQnRZiVMU9vKhZleRGO2MqMtnIHQc5Gc9s0g0h2kXZc27RNG0omywXCnB6jOR6YpTrEiSKLaCCKFWYmJQxWTcMHdkk8jjrx2xUb6pIwKxwxRR+U0SxoDhQxyTySc/UmmvbA/Yp/1/XcsWuk28z3Ik1C3xHB5iMm8g9OT8ucDuODWZIoSRlWRZADgOucH6ZANS2l21nKzqiSK6FHR84ZT24IP5GopHDyMyxrGCchFzgfTJJrWKkpO70MpOLirLU3dI0KOWE3GoTQASWc80FsS/mPtRsPwMAbl7kZx0NRzeE9Sg0X+0pFAjEaysvlyDCNjB3Fdh6jgMTz04OI7fxHcW9mkAtbWR0gktlndGMixvuyv3sfxHBxn3xxUF1q3221VLiytmuFRUF384kIXgZG7aeABnbn8ea2fL/Xz/4Bkr9Rb3TP7KZRc3NrLcIwEtmvmFk7kMdoX2OGzzVvUtNhn1qyt7CFLT7XbQybFEkihmXJwPmc89uaq32sDUZFmudPtDcFw8s6+YrTYHO4B9oz32gGrL+Jna+t7qPTbKJ4IjBhfNIePYU2nLnseowfejS3zFr+As/hS8tdQNrczQQgWxuTLKJEXYDgnayhgc5425PbPGWWfhuXUL+S1s7y3n2IsgkhjmlVgfZIywx0O4Co7jX558KltbQRi1NoscSthUL7u7E5yepNRadq76fbzQfZbe5ildJCs4b5WXO0jaR/ePByPahWv5f1+g9bF+Hw5CNP1Fr/AFGC1urKdYWRxIQpywOdqHOccYPrmqsVjAfDN3dDyJpo5o13CSQPEDu/h27WDY65yMdOaBr0rXV/Lc2ttcpfyeZNDIHCbtxII2sCMZPfvTbbWVttLnsf7Ms5EnILyOZdxIzt6OBxk9vrml0DqJJodzGZSXiMcdulx5gLbXV8bQOM5JbGPY+lWdW8Kajo1j9qu1GwOI3HlyLsYg8ZZQG6HlSw9+RVN9YuX0mHT/kWOJwwkUEOcEkAnPQFmI+v0xLc38es3I+0w2NlNIxeW8xL85xyWUFhyf7qjn0FN26Br1GaFZxX+qfZ5wpVoJmBd9oDLEzKSewBANWI/Dc817bwQXUEyXMRlhmhjmdXAJBG1UL5BB6rTIfJ0WZbu21DT9Rba0bQBJxuV1KnOVTsT0OacviOVZCPsVp9lNv9nFph/LCFt553b87hnO79KNP6+f8AwA1HnwteJfXttcSxQCyjWSaV0kICsAQdqoWHB7qMd8UaV4U1DWYHmsirRiUxI3lysJGGO6odo5HL7Rz7HDB4ikOrHUZbG1ecLGI8NLGItihQV2OpHAHrTDr0kySpf2VrerJO86iUOvlu33tuxl4OBwcjijQNTLZSjlWGGU4IpKKKkYUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAE6WUz2jXICCIEjLyKpYjBIAJyeo6VBWrd6gZNBtIf9HLb5A6rDGGUfLg8DIzzz3qWa7tG1VLaWG1+wiRCXjiAIGBn5l5IPOevtWCqT6rv+Bu4Q6Pt+Ji0V0SvBGjNfDTmuVSUxiERlSNny52/KTu6A8+tVfED2jyJ9jFuAJJF/cBR8vGM4/GlGtzSUbDlR5YOVzHorc8M+X51wskVqS6qomuJLf9zz1CTfK/0GD7jNbPh+3s2vLeG3k0y53aiyXTXEUZaSLKhPLRgSAfm+4OMjOAK6uU5XKxyg025bTG1BVja3VgjETIWUnOMpncM4POKq10FjaFvC+pgT2iNNLE0cb3cSOQhfd8pYHuO3PbNX7iPTF8ISoXs5ZltonhlWSASGQsu5dqoJMgEg72OcZx6K2lyutjmr3TbnT1he5WPZOu+No5kkDDODypPftUVvbS3UpjgXe4RnIyB8qqWJ59ga6G5tII7LQRf3Fq9vAxS6WG7jkZVaUk8IxP3T2qVZ3tNeEt3/ZP2dYboQi1MG1gYmAVvLO7ngDdyc+uadv1Fc5Siuws59LnuLa4uIrJrt9PJEamGCMzCUjncpjVtg/iXB+uDVi2utJSS5aay0yGQyMDHJJDOMC3Y5DKAvLhfugDJwMZxQ42C5w9FdiRpl1cXVwILQx6bILo+WihJkZB+744I8wKMdg59K493MkjOQAWJJCgAD6AdKTVgTuSG2lFoLop+5ZzGGyPvAZxjr3qKtlYVk0W3aU2QdJwxxJEHMeB1AOTzng81ps+nT3EUUcdkc3AAEaJkj7RgDjr8n5jmuR1+Xpc6lQur3tocnRXRH7L5kS3q2K3G+TyhEU8tRt+QOV4I3f3iT68VE0lvE4edbJrpbZy4RUMe7Py8L8pOPT8e9V7e/QHQt1MRI3k3eWjNtG5sDOB6mm1taZqTPcXIK2cLzW5Vd0MaqzDGM5GBnHTgE1k3G/7TJ5vl793Plbdv4beMfStIyk5NNGUoxUVJMlt9Nvru3kntLK4nhi/1kkUTMqcZ5IGBxVat69i/tLQ9NeyuLYJaW7RywS3KRsj7ixYKxBbcCOVz0x2q5efYRpc4j/s77CbVPsuzy/tPnfLndj951353fLjp2rZrcyT2OfvdNudPWF7lY9k6742jmSQMM4PKk9+1Va6s20MEfhw3s9jJDby7blFuopdoMxblVYkjb14+tSW1vp+maegun0ue6VLt8CSOYA7U8sEjIPIJA56keop8uornIUV2+q3mkwy3s9lDpTPHFMsCrFGy586MKdvRjtLEZzkZ7cVh6laxan4hEWnG1j823jkPl4WMP5QZwNvA53cdj6VEmox5mVG8nZf1t/mYlW4tJ1Geza7g0+6ktlBLTJCxQAdfmAxxVStq4vrZfDmn25tbW4m2SgyO774cuccK4HvyDTEZ82m3MFhDeyLGbeZiqMkyPyBnBAJKnBHBxVWuhW0UeEbYTz2pVbwzyRpdxGTyyqjO3dnPB4xn2q9eR2MknlTzaVBA17GLKW2WNmSHJyXC8kY25EnOfxp8utv66f18hX0uclHG8sixxIzu5CqqjJYnoAKRlKMVYFWBwQRyDXa317YWGo6NeQLYxypdMLgRm3nxGChBPlIqA/eIIGR68YGHrENze6+sMklhmaTbE9qYdhBbgt5Xf681MrJXKV2zFqzaabfX6u1jZXFysf3zDEzhfrgcdKhmiaCeSJyC0bFSVORkHFdBoapeaclnqgtf7OExfzTexwzQE43Mqk5YYA42nPYg01qtAleLszn/JkEImMb+UW2h9p2luuM+vNMrp4L6Sbwi9naSacWt7hyRcRW6O0RQYZd4yW4PIy3T2qnqOoxR2cFna29i0clnH5jiBDIJMAk7x8wbjGM49RQ/wDL8riX9feZc9lcW1vbzzx7Y7lC8TbgdwBIJ46cg9agrrZLaA3Hh+e8Ok4SVVvBDLb4I83jcqHkbepx9afprWeqzadJLb2QlSS68xIYY1CosYMZZeFIBzy5wccnrT5dRXOPoro/ENs1zqWnW8MVtGksYjS4jjhRZmLEFj5JZRjgYyTxnvXPzRNBPJE5BaNipKnIyDiour2Ls7XLFtpWoXlu9xaWFzPDGSHlihZlXAyckDA4qpXXeG7iGHS7J3+w5t76SSSWe6EctupWP50XcNx4PG1uR0plv/Z40+EodPNgbeT7V5/l/afO+bGM/vP7mNvy+verasSmcpVq902509YXuVj2TrvjaOZJAwzg8qT37V0F59hGlziP+zvsJtU+y7PL+0+d8ud2P3nXfnd8uOnakktLaO38PjUp7R7aBit0kV3HIyq0pbojE4weSOn1o5dbCvpc5aiuyuDYza1YJJBY2gVpGMyXVpKsoC/KhCRhFyRgM6n73tik1G40y2nlubeHTjP/AGYD5ZMMyrN54HRFVCwTnhcEeoySraXHfWxx1FbepWsWp+IRFpxtY/Nt45D5eFjD+UGcDbwOd3HY+lYlTdX5SrOyl3Lkej6nNZG8h067ktQCxnWBigA6ndjHGDVOtu0tGfwleHzrRWknilWNruJXKosob5C27+IcYyc8Zq7cXli2n3UCRaeohsLZ4WWKPe037vf83Unlsr04ORnJq2ibnNSQyQlRNG8ZZQyhlIyp6EexpldffXsOoXGjz376e2m7YFuBEkCSqRwwKqBIF4PQYxjHarGyzu9esom0+zUIsztLFcW0ysAhKlljVEABGfnxnoTijl1+Yr6HEUV3L/Yk1SM3NrZnFkfssgFpDHcSb/mJKh4gQuRhs9OxIrMuL21t31SeCysIbgeQIo2MNyoPO9lwNhz3AGBnoMClaw7nM0VLcz/abqSfyo4fMYtsiXaq/Qdh7VFSGFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUVryWlpbaDHP5lrPNMzgE+aGXG3heAMjJznj0zUMmjSpKIUuLeW4LKvkIx3fN06gD9eO9ZKrFmrpSRnUVpJory/NBd20sQDbpVZgqbRk5yoPTngc1FqOly6YyrNJFJuLL+7JOCMZHIHqKaqwb5U9ROlNR5mtClUtrdTWV3Fc2z7JoXDo2AcEdDg8Va0zSJdUS5kSaOCK2QPLJIrsFGfRFY/jjA9asad4el1Hy9t7awCecwW5mL/v2GM7Qqkgcjk46/WtbMybXUyCckk9aK14ILaXwxfSPaxi5tZogs6s+5gxbII3bew7ChvDd4NAOroyvbqoZh5Uq4Bbb95kCHk4+VjRYfWzMiitrULOK5tdGbT7JIZ71GUxxMxDuJCo+8xxnjvinWOi251cWkl7aXZMM+4RNIoidY2ILMyqMZHUEjj0osK5h0VuQ+FLy5vI4bSaG4jkt/tKzwpI67N237oTfncMY2+/Tmph4L1Dz5Ypp7WAxnH75nXd+7MnTbkfKCfmAxjBwaLMLoxvt9x/Z/2EOq25feyrGoLkZxuYDLYycZJxniq1bMvhi8juoYVmt5TLcfZw6M21WwGycgYBVs59j6VjuoSRlDBwCQGXOD7jND8xiU+KV4JkliO142DKcZwQcirps0OkQSR+U0kk+wuHbK8D5SpGPfIJ61PJ4cuYyo+0WzM0gjwrNwd+w9uzf/WzWDqw2kaqlN6oyCSzEnqetFaJ0WYlTFPbyoWZXkRjtjKjLZyB0HORnPbNINIdpF2XNu0TRtKJssFwpweozkemKftYdw9lO+xn0Vq2uk28z3Ik1C3xHB5iMm8g9OT8ucDuODWZIoSRlWRZADgOucH6ZANVGak7ImUHFXY2it6PSLMeG4r542kmlSQ5OoQwhSpIGI2Xc/4HnpVabQJ4bWSRrm2M8USzS2gZvMjQ4wTxt/iBwGJGenWtGrGdzKorYn8NzxXC2sd7Zz3jSpF9midt4LdDyoBHTOCcZ5xT9Q8M3Oky2x1CWOOCdyolkimjAIwSCrIH7jkKR+tHKwujEp8c8sSusUroJF2uFYjcPQ+orYvdF8zxFqNtD9nsYLXLuWkeRIkyB127m5YD7ueaV/C8kU06y6nYJHD5W6Vmk2kyqWUAbNx4Hpxn64OW6HzWMOiuiTw0lvYy3WpNKkVndTQXZiIOSgTaqnH3mZiMnjAzjjmG28J6he6UdRt0xCUeSNWSQlkXOTvCbB0PBYE46cjJbS4rmHRWlLodxCJmeWHy4rZLnzMna6vjaF4znJxj2PpWbS2GFPhnlt5N8ErxPgjcjEHH4UyijfcNtQooooAKKKKACrFlfXGn3QuLRwkgUr8yBgQRggqQQQQehFV6KALtxq99czRytP5bRoUjECLEqKckgKgAGcnPHNUqKKOtw6WCiiigAooooAKKKKAHxzyxK6xSugkXa4ViNw9D6imUUUAFFFFABViyvrjT7oXFo4SQKV+ZAwIIwQVIIIIPQiq9FAFm91G51BozcsmI12okcaxogznhVAA5JPSq1FFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAE0ly8lpDbsF2QszKQOTuxnP5U9r+c6gL1SqTKwYFRwCPr9KallM9o1yAgiBIy8iqWIwSACcnqOlQVCUHsW3JbmgdYkVGSC2t4I2VwyRhsEsME8knOOnOPaor7UZtQYGZUXDs/wAgI5bGe/tVSikqcE7pDdSbVm9C7pmojTLgTraQzyqQ0bSPIpjI7go6/rWvo/icx6vbSanFbPEL03JmaNi0BYjeVCnHOOhBx2wa5uitbsyaTNS31lbbTJ7EadaTJOwLyyGXeSM7Tw4HGT2+uafN4jnm0+S1Npaq0sEdvJOqtvZEKlf4sD7o6AZqkNNuW0xtQVY2t1YIxEyFlJzjKZ3DODziqtGuw93c15vEDvDZpb2FpavYsGgliMhZDu3fxOQeeeRUcmtZuzc2+n2ltKySo5i8zD+YpUnDOQMAnGMCq17ptzp6wvcrHsnXfG0cySBhnB5Unv2qK3tpbqUxwLvcIzkZA+VVLE8+wNGotC/Dr1xFHFE8EE0CWxtmhkDbZE3l+cEHIY5BBHT65dB4gmtFkS0tLaGJ3Z/LXeQpaNozgliejE8nr7cVk0UXbCx0Ka4E03ULiSaJru+VYkt40b9zhSpck8ZKEqME/ePTFc9RRQ3dgtC8mqbLFLZbO3wj+YHy+7d0z97Hb0xVmDXZXvYTciNIvODOyqeB5vmHv6/pWabaUWguin7lnMYbI+8BnGOveoqwdKnI29pOJonWJEkUW0EEUKsxMShism4YO7JJ5HHXjtio31SRgVjhiij8poljQHChjknkk5+pNUqKr2UF0B1ZvqT2l21nKzqiSK6FHR84ZT24IP5GopHDyMyxrGCchFzgfTJJoSN5N3lozbRubAzgepptVZXuRd2sTzXkk9pbW7hQluGCEDk7myc1en1+ee1dGtrZZ5YlhlulVvMkQYwDk7ew5ABOOvXNOLTb6eze7gsriS2jzvmSJii465YDAqtV67EeZdfVrltZGpoVjuRIJAUHAI9jn0p02o28t1HMNJsowpJaJGm2yZ9cyZGP9kiqFFF2Fjq9N8SQzave39wtrp11NBtR0a4WN33qTuKMXXgH7pA9az9Z1SF7meOwKyRzvFNNJmRg0qKwJUud20lz97n6dKxKKOZhY2D4mvWuXlMcBEk800sZU7JPNChkIzyvyjHcdc5xiB9XE1ittcWFrN5askMrGQPCpJO0YcAgEnG4HrWdRR0sM2dS1CJdBs9Lt7hLkxkySzIrAdTtQbgCQu5j06sfTNY1FFJ6u4bIKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooA1bvUDJoNpD/o5bfIHVYYwyj5cHgZGeee9SzXdo2qpbSw2v2ESIS8cQBAwM/MvJB5z19qxaKx9jH8/wATb20vy/A6JXgjRmvhpzXKpKYxCIypGz5c7flJ3dAefWqviB7R5E+xi3AEki/uAo+XjGcfjWPRUxo8slK5Uq14ONjc8M+X51wskVqS6qomuJLf9zz1CTfK/wBBg+4zWz4ft7Nry3ht5NMud2osl01xFGWkiyoTy0YEgH5vuDjIzgCuKqW1uprK7iubZ9k0Lh0bAOCOhweK6kzlaumbdjaFvC+pgT2iNNLE0cb3cSOQhfd8pYHuO3PbNX7iPTF8ISoXs5ZltonhlWSASGQsu5dqoJMgEg72OcZx6cgTkknrRSvpYp/Fc6a5tII7LQRf3Fq9vAxS6WG7jkZVaUk8IxP3T2qVZ3tNeEt3/ZP2dYboQi1MG1gYmAVvLO7ngDdyc+ua5SincVjsLOfS57i2uLiKya7fTyRGphgjMwlI53KY1bYP4lwfrg1YtrrSUkuWmstMhkMjAxySQzjAt2OQygLy4X7oAycDGcVw9FDkFjsSNMuri6uBBaGPTZBdHy0UJMjIP3fHBHmBRjsHPpXHu5kkZyACxJIUAAfQDpVj7fcf2f8AYQ6rbl97KsaguRnG5gMtjJxknGeKrUm7sFobKwrJotu0psg6ThjiSIOY8DqAcnnPB5rTZ9OnuIoo47I5uAAI0TJH2jAHHX5PzHNcnT4pXgmSWI7XjYMpxnBByK5JUHLqdUa/KrWN8/ZfMiW9WxW43yeUIinlqNvyByvBG7+8SfXiomkt4nDzrZNdLbOXCKhj3Z+XhflJx6fj3rCJLMSep60U1Q8wdfXRG1pmpM9xcgrZwvNblV3QxqrMMYzkYGcdOATWTcb/ALTJ5vl793Plbdv4beMfSo6K1jTUZNoylUco8rOktUivtDjTWTaxw20D/Z7mO8jEyfeIRotxLAsf7oPOc4qzeyaUugRpb2dtJC0MWZhdQCWN+N58vZ5pOdwwWK85HGMclRW3MZWO6njsUCyzLpRsY9Wt1hMEcbFbf5/vsoycgDIck8HIqnZ2el2X9mxajJYSy+dcs4injcA7F8oO/wAygbuRnKjPI6iuXW6mWze0V8QSOsjJgcsoIBz16MfzqGjm/r7hW/r7/wDM6y6vLOG6vJvsGnxTpYjyw0tvcK8nmrhgI1CbgueAOgyc5NKxspLy+k0/+zBetDbNCswiEPKAzbQ/7sNuxwenzYrkqKLjt/X3HX6Q9jHYzNcWljd3v2p/tMb3NrCpTAwELoRtzu/1ZUj/AL5xyLkF2KjaM8DOcUlFJ6jWgUUUUgCiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKANeS0tLbQY5/MtZ5pmcAnzQy428LwBkZOc8emahk0aVJRClxby3BZV8hGO75unUAfrx3qpJcvJaQ27BdkLMykDk7sZz+VPa/nOoC9UqkysGBUcAj6/SsFGouvf/AIBu5QfTt/wSymivL80F3bSxANulVmCptGTnKg9OeBzUWo6XLpjKs0kUm4sv7sk4Ixkcgeop51iRUZILa3gjZXDJGGwSwwTySc46c49qivtRm1BgZlRcOz/ICOWxnv7Uo+15lfYcvZcjtuSaZpEuqJcyJNHBFbIHlkkV2CjPoisfxxgetWNO8PS6j5e29tYBPOYLczF/37DGdoVSQORycdfrVXTNRGmXAnW0hnlUho2keRTGR3BR1/WtfR/E5j1e2k1OK2eIXpuTM0bFoCxG8qFOOcdCDjtg11Kxyu9mUoILaXwxfSPaxi5tZogs6s+5gxbII3bew7ChvDd4NAOroyvbqoZh5Uq4Bbb95kCHk4+VjTLfWVttMnsRp1pMk7AvLIZd5IztPDgcZPb65p83iOebT5LU2lqrSwR28k6q29kQqV/iwPujoBmlpYp/F5D9Qs4rm10ZtPskhnvUZTHEzEO4kKj7zHGeO+KdY6LbnVxaSXtpdkwz7hE0iiJ1jYgszKoxkdQSOPSopvEDvDZpb2FpavYsGgliMhZDu3fxOQeeeRUcmtZuzc2+n2ltKySo5i8zD+YpUnDOQMAnGMCnp+YtSzD4UvLm8jhtJobiOS3+0rPCkjrs3bfuhN+dwxjb79OamHgvUPPlimntYDGcfvmdd37sydNuR8oJ+YDGMHBqjDr1xFHFE8EE0CWxtmhkDbZE3l+cEHIY5BBHT65dB4gmtFkS0tLaGJ3Z/LXeQpaNozgliejE8nr7cUO3QNR8vhi8juoYVmt5TLcfZw6M21WwGycgYBVs59j6VjuoSRlDBwCQGXOD7jNdAmuBNN1C4kmia7vlWJLeNG/c4UqXJPGShKjBP3j0xWOl5s02W0+zW7eY6v57JmRcZ4Ddgc8ik7dAV+pcm0CeG1kka5tjPFEs0toGbzI0OME8bf4gcBiRnp1qc+FpjerZJqNg92ZAhgEjBlyCQclQCOOcE4zzioJ9fnntXRra2WeWJYZbpVbzJEGMA5O3sOQATjr1zEutXCa6NVCReeH37cHbnGOmc/rT0uGtiXVPD82lWkVw91bXEchXHksxIDLuUnKjqPxqppsMdxqMEcxjCs4BWQsA/P3cqCRn1qa91q4v7NLaZIlRPLwVBz8ibB39B+dVrO5+yXST+THMUOVWTdgEHIPBFZz2fKVDpcuQaNLeyzmKSCFY5WjCu56gFsA49FPJ9KUaFLv2vd2qbpREhZm+diARj5ehDDk498U0a1KjuYbeCIO5kZVDEZKsp6sezGphrIFpG7QQSXSzl13hv3eEQKw5wfu989Olcz9vfyOr9y/XUrJo8zIN8sMczqzJA5O9wM5IwMDoepGcUjaZst1lku7ZGeMSrEWbcVP4Yz7ZzSpq8yxruiikmRWVLhwS6A5yOuD1PUHGaqz3D3Hl7wo8uMRjHoK0Sqt6mbdJLT+tv+D+Bc1PToLJIWhvI5S8SOUAfJyOoyoGPqc1X0+xk1K+jtIWjR5M4aRsKMAnk/hRPem4tYopIIi8ShFmG7dtHQdcd/Sm2V5JYXa3EIVnUMAGHHKkH+daU1JK0jOo03eJpWvhqe/vBBp13bXi+UZXkgWVwgDbcFAm/OSOinrnpkizY6MLHxFJpWr2kU5aF3DMZFKfuy6sBlSD04YfgKyNP1F9PaXEUVxFOnlywzA7XGQR0IIwQDkEdKs2OtLp+oSXcGmWZZlKpGzSlYwVKnHz55BPUn2xWuhnqP0Xw3d69HIbFl3IwXYYpWyT0+ZUKr/wIinjRLT/AIRpNQbU4I52mZDGwk7KDt4Q/Nz1zt5HPWix8TSafs8jTrIiG4NxAHEh8liADj5+RhR97PtVSDVTFp0tlNawXMTyGRfN3gxORgldrDtjrkcCh2toCvfX+t/+ATzafGPDtnPbrDNNNctG0kbvvB2qRGVKgcZzkE5zUh8MXBnSGK7tJpftCW0yI7ZgdjgBsqARkEZXcOPpUI1zbpEdhHp9qgjfzVnVpfMD4A3/AH8Z4HbHtUzeJ7j7Qk8NpaQS/aEuZmjRv37qcgtljgZJOF2jn6UaX/ry/wCCLWwn/CNXDXFnFbXVrc/a5HiV4WYhXTG4H5cnGR90HPbNPtNGgabU7ea7t3a3tt4nPmIkTCVFOQyhsgEjG09eMmqKamwtre3ntoJ4YJZJQkm4bi4UEEqwOPlGMYqe88QXN6Zd8UKI9qtqFUu2xFcOMFmJJyO5PHHpg0/r0/zH1/rv/kWrHQ1i1yWzvHtbhktzJGhkkRZgYy6srKhPAwcNj0qrp2gS6jaxTLeWlv585t4kmZg0kgCnAwp/vDk4HqRxT7fxFLBqX257G0nmEKwr5nmAKqx+XwFccleuc+2Kng16Cz0uA29jaG6jvJZo1YSkW+VjClctg8qfvbun5nu/16i1/r0/zMF0aORkcYZSQR6GkpWZncs5JZjkk9zSVJQUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUV0194fsoNDe4i84Tx2sE+77VHJuLhSwMSrujA3feY44HqKp3fhW+tLK3utyyRXEiRIfLlj+Zhlf9Yigjg8jI/Sq5Xewrq1zForYfw1cjW00qCeKe6O7eqRygx4GTkMgJ45+UHPbPFQaxol1olxHFd4PmpvRgjpkZI+66qw5B6gVPS4/IzqKtWenTXwfyHtl2Yz591HD+W9hn8Kgmha3meJyhZDgmNw6/gykg/gaAGUVv6RoUcsJuNQmgAks55oLYl/MfajYfgYA3L3Izjoajm8J6lBov9pSKBGI1lZfLkGEbGDuK7D1HAYnnpwcNpoV0zEorRvdM/splFzc2stwjAS2a+YWTuQx2hfY4bPNW9S02GfWrK3sIUtPtdtDJsUSSKGZcnA+Zzz25osFzDorcn8KXlrqBtbmaCEC2NyZZRIi7AcE7WUMDnPG3J7Z4yyz8Ny6hfyWtneW8+xFkEkMc0qsD7JGWGOh3AUWd7BdGNRW/D4chGn6i1/qMFrdWU6wsjiQhTlgc7UOc44wfXNVYrGA+Gbu6HkTTRzRruEkgeIHd/Dt2sGx1zkY6c0DMqitKTQ7mMykvEY47dLjzAW2ur42gcZyS2Mex9Ks6t4U1HRrH7VdqNgcRuPLkXYxB4yygN0PKlh78iizQroxKK0dCs4r/AFT7POFKtBMwLvtAZYmZST2AIBqxH4bnmvbeCC6gmS5iMsM0Mczq4BII2qhfIIPVaLMLoxqK2z4WvEvr22uJYoBZRrJNK6SEBWAIO1ULDg91GO+KNK8KahrMDzWRVoxKYkby5WEjDHdUO0cjl9o59jgswujEorSSzit9Emu7pd80kxt4E3YClQC7nHXGVAHT5j6Uj2kE3h9L23QpLby+TcDcSG3Asj+3RgfoPU0DM6iiikAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQBqz6/LNaGJbS1ikaGOCS4QOXkRAoAOWKj7q5wB0+tSX/iSXUI5ElsbNBNcLcT7A4MrgEcktwDuPAx7YrGop3YrGy3ia532nk21vDBaq6JbgyOjK4w6nexOCOwIx1GDzWbd3EdzMHhtIbRcY8uEuQff52Y/rUFFK9xlqz1S/wBODjT765tQ+N/kTMm7HTODzUE08tzM81xK8srnLPIxZmPqSetMooA17fxHcW9mkAtbWR0gktlndGMixvuyv3sfxHBxn3xxUF1q3221VLiytmuFRUF384kIXgZG7aeABnbn8eaz6Kd2KxpX2sDUZFmudPtDcFw8s6+YrTYHO4B9oz32gGrL+Jna+t7qPTbKJ4IjBhfNIePYU2nLnseowfesSii4WNW41+efCpbW0EYtTaLHErYVC+7uxOcnqTUWnau+n280H2W3uYpXSQrOG+VlztI2kf3jwcj2rPoou73Cy2NUa9K11fy3NrbXKX8nmTQyBwm7cSCNrAjGT370221lbbS57H+zLORJyC8jmXcSM7ejgcZPb65rMopdLDL76xcvpMOn/IscThhIoIc4JIBOegLMR9fpiW5v49ZuR9phsbKaRi8t5iX5zjksoLDk/wB1Rz6CsuinfuKxrQ+Tosy3dtqGn6i21o2gCTjcrqVOcqnYnoc05fEcqyEfYrT7Kbf7OLTD+WELbzzu353DOd36Vj0UXYWNgeIpDqx1GWxtXnCxiPDSxiLYoUFdjqRwB60w69JMkqX9la3qyTvOolDr5bt97bsZeDgcHI4rKoouwsaMV1BLoMtjO3lyQyme3bBIYkBWQ+mQFIP+yfWiS7gh8PpZW7l5biUTXDbSAu0EIg9erE9uR6VnUUXGFFFFIAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAqe4s5rVInmCbZRlGSRXBH4E1eugBpkX2P7GYPJXzc+X52/+L73z9fTjH41LNbD7FpaSXNsoRispE0cmwM+clQTnj/69Ye129Tf2Wj9DForo742fnWLxw2ryrM4eMTwgMoA27iqhB36g+/pULCKPWkbFrcboSSm6CNYjyMZ5jYj6c+g7Sq91e39f11KlQs7XMKrdnpOo6hGz2FhdXSKcM0MLOAfQkCk1NFTUJBHNHMODvjRUXOBkYX5eOnBIrQtb62tvDHlTWtrdyG7ZhHM7goNg+YBGU/nkV0RaceY55JxlymL060V1+lR6YPDLLdvZyGS2nYl5IFeKQBtiYKeaTkAghgOcexoXxEej27aZ/ZptBBGZdwgNx5ufnyG/efe9Pl2+2atqxN7mFJDJCVE0bxllDKGUjKnoR7GmV199ew6hcaPPfvp7abtgW4ESQJKpHDAqoEgXg9BjGMdqW7axm1K3hltNPg3R3CLMLu2kRiYyEz5Sqq4bBDMM89eOBx3+YJnH1NaWV1fzeTY201zLjOyGMu2PXAroytjpwhGdOnni0ti2Ckq+f5pxnszAY9cgdxUMF2NY0S+tozZWd7JLDIyhkto50UFSOSEByQ2OM9R0otr/XewXMlNHvnhupPJCfZM+fHJIqSJjAPyEhupHQVDHZXEtjNeJHm3hdUkfcPlLZwMdexrZ0OxdY9WSS4so2a2aBd97Cu59ynAJbkYB5HHvS2lqZvCd5HI2miVZUaAtNbrNgFt/OQ5HTg9e1Jr8v1C/wCZz1FdxfT6M2m3EMSaWp2OqMiR7xiKIjBHOd5f34I7YqHXX0+10lXtDYS3VvdIIZE+zOZY9rZYxogwCQOH3Ee3duNmJO5yMkEsU5hljaOUHaUcbSD7g068s59PvJbS7Ty54W2umQcH6jiuk1+4ub3xElzavpUkUrl7dv8ARRkFQf3mf/anepZ7W0h8Rao9s2lF5Iy9gGmgaDO5QeM7Adu7AbH06UWHc5Ciuw1OwtJ7y7gR9OgmzZShyUhQp5J8wrkDgllJUDJ9KsRSaPHPqDWtpY3Z+3y7o3ubeFTDkbAhkQ/L977hUj8sHL/X3Bc4yG1mnimkiTKQLukbIAUZwOvuelTS6VqFvZpdz2FzFbPgrM8LBGz0wxGDmrshz4QBgAVP7QbzgDkj5B5ef/ImPxrYv7i3/sWfmyjWawtoxNDch5pXVY8o8e47QMH+FfujnnBLBfX+u9jj6K7PWo9LXTYkhFjPNFeRLGyT26meLDbiRGq7ASB98kj27tmXTTr1lNePZCN1lxaxm22xNt+TdLECjAt3ZeMcgjknLqK+hylraT3s4htYzJKQWCA8nAycepwOnWoa6DVJGk8QWP8AZtnDbXQKbfs9xDLvfd8p/cqqA9BgAdMnrQ3kf8JhqX2P+z/I3XGz7b/qsYb7vv8A3ffFK36/gM5+iiikMKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKtQ6hNBEETYdvCsyAlQeq89j6fX1OatbOnaNaXuh3d7PqcFtJDJGoWQSEKGz97ah644wfXNFridupT/ALUnH3UhXHCYjHyL3UZ7Hvn39Tk/tScfdSFccJiMfIvdRnse+ff1Obdn4au7/SZtQtXV44VdmXypeijJ+fZs6c4LZ/HiptR0FVhM9lNb/u7KC4ktg7GQBkTc/TH3m6Zz7YxRy6CtG9jO/tScfdSFccJiMfIvdRnse+ff1OT+1Jx91IVxwmIx8i91Gex759/U5vXthHeQaL/ZtlHDcXyshSN2IdhIVB+djjp64qWfwbqNvdW8UkkSLOJCssqyRKNi7myJEU9O+Me/WjlFaJmf2pOPupCuOExGPkXuoz2PfPv6nJ/ak4+6kK44TEY+Re6jPY98+/qc3E8Nzyahb2sN1BMLmEzQywxzOrqCQflVC+cqf4e1WF8GX/2ieGW4tYWhbafNZ13fu/M4G3I+UE8gYxg4NHL5BaJl/wBqTj7qQrjhMRj5F7qM9j3z7+pyf2pOPupCuOExGPkXuoz2PfPv6nOrb+CNUupJlhMbiOTyldY5WWRsA8FUO0YYcvtHPscc8ylHKsMMpwRQ423HyxZc/tScfdSFccJiMfIvdRnse+ff1OT+1Jx91IVxwmIx8i91Gex759/U5pU+KJppkjQqGcgAu4UfiTgD6mlZByxLX9qTj7qQrjhMRj5F7qM9j3z7+pyf2pOPupCuOExGPkXuoz2PfPv6nLbzTZ7FVad7VgxwPIu4pj+IRjj8agt2iW5ja4UvEHBdR1K55FFkHKuxdn1y8uJPMl8ouqqiHyx8iAY2D2wAOff1OY/7UnH3UhXHCYjHyL3UZ7Hvn39Tl9jawXK3MjqxWMjYvnrFwSerMCKLXSZb+SUWuBsfaFIZ/p8yrt/E4FTddRWiVkvJYxcLCRHHcDbJGBlSM5Awc9D0PWoKsizItRNLNFEGBKI24s+OOMA9+OcVLcWEMVjBMl3GzyKSUw3zHdjj5f5mquitCjRWnLoN5CYg+0GSRYjlWAVj05IwfquarQ6fLOISjRgSu6LuOMFQCc/nS5kwuiO1u57KbzrWQxybSocDkZGDg9jg9RzUNX47CILJNJcLPBFHvPkEgkltoX5hx9cdKLFbWe/MX2bMTo2BJISykKTkFcDqO4o5guUKKt2cEcttdO3ls6R5VSzBl5HzDAwfTBNTRaLNLbrKJoF3KGCljnnOO3faaHJLcLozqKvf2VN/Z/2wMvlYBJKOMAnHUrg8+hNF9YQ2vleXdxvvRGIw+eR1+6OP1o5kF0UaK0Lix3X0cNuI1VoVcuHYrjbkscgEeuMVFLpzxIXWaKVPJ84Mm7ld+3uAc5o5kFypRWhDotzPI6rjCKjFgrt94ZAwqk9PbFQGNLSW5gvIy0qqUTB+6+Rz16YzRzILorUUUVQwooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKtWdpDdb/P1C3s9uMeesh3fTYjfriqtFAGn/ZVn/wBDBp3/AH7uf/jVWbOS306O4gN1Y39tOFEqukwXcDlSMbG4ycn68HjONHDJKHMUbuI13OVUnavqfQcimUCaudLBq32axS33acWigkthMUdnRZN2VXDYI+Y/NjIwevGWT6kstm0O+wgkaCO2nuIxIZHVQAq8sVx8q5ZQOh9s87RRr3FyvudLJqqeXZCAadaSWDfupYfNZkbdux8zsGUk8nHr7Zha6t0uvPt106zcRyRSiHzWBMilejOcgAnkYA54PGcCrN1pt9YJG99ZXFssn3GmiZA/0yOaNQ5fM2bfUvs8aRTS2M0MdqbOSN1chgZPMHKsCfmP3lIHHQ8ZnfXpmkdmnsEz8rqkbYU+SYQFAPK7T1HQj6A89Y2M+pXa21oIzK/3VklWPcfQFiBn2qGRGikaNxhlJVh6EUO/V7iUfM6KTVEuEZNR/s66QPuCuJf3TlQvylHBIIUZPI4OB0znLpVqVG/XNPjbuhWc7T6ZWIg/gSKzKKPUaVuppPplokbMuu6e5AJCKlxlvYZiA/WqEUskEySwSNHIhDK6MQVPqCOlOW1me0e5CfuUcIXJA+Y9APU8dqnu9H1OwhE19p13bRMdoeaBkUn0yRT8x+Ql5quoaiqrqF/dXSocqJ5mcKfbJqpTo43lkWOJGd3IVVUZLE9ABSMpRirAqwOCCOQaQy3aqrWkqyXMUSSMAwYEtkdOB25OT2x9M3rSVLKNF+0Wb+TJvBdWYo5x0wfmHAyecYOO2cWipcbkuNzWxD9lEFxPaSCMFQ5D7oyecLtI3DPOecc9eMpth+yLBLc2x8okCT5i0bE5wuDhh74OOfbOVRRyhyvubEjxNcJcGezjnjkDPKgckv1HGcEZ6kDsfbLmmT90Fns4VgL5WNWIDOMepLD37YPXjOLRS5ULl8zTto47RnzeW7oy+XNGwLKSeg45IzzuHTH0zLAba1ummV7IEDYV/eMEYjHy4bLA55POMHHbOSiNJIqIMsxAA9TSMpRirDBU4Ip2TY+R2ubEDW1t53zWREgKSA+YducfKuG5Xj73sevGXJc+Wqr9rtVEYVSArHBG7AHPI+Y5Pt9M4lFLlTFy+ZtPMjWzRPcWalo1hlkVWLfLjaBzgjgZIHY9eMxyLBJFEs89oWhARnG8k46LwcFfVgO30zk0qqXcKoyWOAKfKkNRfRm208XnRyeZZRmNPJcKHYEEYC/eOVx/EPQ9eMrJNb/uv39nsWDyJYsOVB3lgBg5I6fMCeh65GcNlKMVYYKnBFJS5UHI0bbzpJJIbmezdGCq8W1tqsowu3aQSMdSDjrweM5FwgjuHVTGRn/lmSVHsCeuOlR1Zg06+uo/MtrO4mTON0cTMM/UCjSOrZUYu9kVqKVlZGKupVlOCCMEGkqwCigDJAqW6tZrK6kt7qMxyxnDKe3+I96AIqKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAK09GgiuUv4JGt1mktsQG4dUAcSIeGYgA7Q3cenesyimgOzgvILLWL+ysJdMKy6fGkcjJCYnlCR5G9xjkhuDwT15p+nXOnwWIF9HpLXR81pCY4WG7zowuMfLjaXIA4I9QK4miq5tbk8ulixqAjGpXIt9vlec+zZ025OMe2K05rrQm0fy4bbF55YG/wCzOPm7nd9oI/HZ+ArEoqForFPV3NDQbq3svEFjc3v+oimVnOM4APXHfHX8K0bTT1tdcZ9TubC4aQSmCQ3ccsby4O0vhjgE/wB7A9eM1z1FO4rHVJHFJ4h0d2k06K5hZZLx4ZooogBJkcghC23HC/zzUC6fFM2owSXGnrP9rilSR7qIr5X7zdhwSD1XKjn24rnKKfNt/Xb/ACCx3wudEgluRFHpLAzTshaONuPPjCYz22FyB0I9QKoPc6YrCzSLTvIe0u2eTy0L7w0vljf1B4TGCMgjrxXIUUr6JBY1tSz/AMI9o/l48nbLux/z18w7s++3y/wxWl9kjhuvDhvJrCSCFliuFF3FKFzO7HcAx42kZPTmucW6mW0e1D/uXcOUIBG4dCPQ89qip3C11Y6mDV7aOTR78x2EM0d7IsyxW0fywgxlcrjtlsMfm96nheE+Ibt9UGmzSeUxsxDJapGx3j7zbWjDbd33x+RxXH0Uc2wWO3sIrOa61K5NraWYR4EBJtbiP7rFgC5SLLYB+XkdhjNLC9jFJemfS7P7X9sfzrSW4toAkWBsVS6EEfe5jKnv6GuUstWu9Phkht2iaKRgzRzQJKpIzg4cEA8nketVrm5mu7mS4uZGkllYs7t1JNHN2FY2H1KG00ezS1tbCRpVlE4khSRxliFBY/MpA5BBBrLNlINNW93J5bSmLbn5gQM9PTmq9PeeWWNEkld0jGEVmJCj29Kh3drFq2tx1qA15CrAEGRQQe/NXiYhLELhbcMJG4TbgJjoce/rzWajtHIrocMpBB9DSE5JJ61EoczubU6qhG1i3BM013FuS3XafmLIigjPfPH9aswiCGSUSC3bMhxkqw27WPH44rLopSp3HCs46tXZsQfZjMzOYCjMoZSY1CjaMnkE9c8DFUjMkNvGI0hYktu3IGOO3PUVUopKkk9WOVdtWSLr+X9i+0BE3Ooi24HDDq35Y/E1WtyFuYiTgBwST25oeeSQpuIxGMKNoAH4dKc91I6FWWEA/wB2FAfzAqlFpWJlOMmn2Lzm2a8RgYvILtkEjdv56n0zjnp7daY/ly3Sw+RGrSpsBV0b5s8H5QAOePpWdT4pnhJMZAJGM4BI+h7fhU+ytsy/b3eqNewudGhmmXUrdZVXakREDPnGcn5Zo+vXnP4VXE1r52pm22xwyQkRKQV/jQ4ALMex43H61m0Vo43ikYc3vcxpC8S20y2WKG1kd/MEu+JWbrxz1HtjFWWgtBZT3Be12tZxLGgdd/mApuO3qDw3PfPesSis3S6p/wBXuWqtt0butPZ/ZStnbweXvHkypPEWC47qqh/ruJxiodd/49tJ83/j5+wr5n03Nsz77Nv4YrJB2sCMZBzyM1Jc3M15cvcXUhklkOWY9/8APpV04ckOUipPnlzEVFFFWQFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAVreGtNttU1y3t7y4jiiLglH35lHUqCoOPxxWTV/R5zZ6gl6ssUbQHK+aCwJIIHA5x6nt+QLTs7ilsXotG/tzVpodLNpGsaKf9FiupEb3GUZx75wPSq/9gTRG4N/dW1ikE7W5eYuweReoARWPHHOMcirGnXUem280DyWNxFI6M6zK7bJFztI2kbh8xz1HXg8ZnfVFma5/tA6ddRzXDXDxusmIpW67CjAlTxnkj5fplXjb+v6/pE8xVGi2Z8NJqJ1SBJmmZDGyyY4UHbxH97nrnHI560s3hPUoNG/tKRQIxGsrKY5BhGxg7iuw9RwGJ56cHBbzQwabJY3EllPCXL/ALzzCYZCMZQqw3DGMk5Hy9Dxl11cw3lqqXZsDcRosTXYEnmfLwowG2sMAAtt7HqcEu8b/cHMVm8P3SzzxGSHdBerZMdxwXYsARx0+U+/tUOoaYNPUYvrS5YO0bpAzExsPXcoyPcZHHWtq41tphy2mwt9qS6neJHJkmXOCfmOVOTnbgDnA6bslrKG5uJHm1O0gLMTIXDlQxOcDarEj36cfTKutl/W3/BGpd/6/rQd4f0u31bU/s91drbJ5bsM7ssQpPGFPpk57ZxzU2n+GbnVJp10+4iuIYNoaeKGZ1JbOAFEZfseSuOOvIzAjLol7Bd2V/Z3zqTlI0lxgjBDblXggkcH8qItaWITxjTbNrSYqxtWMpRWXIDA79wPJ/ixz9KrTqPUvReC79t4mubO3aN3Vllds/K4Rj8qnuy/XPHeoP8AhF7ny1zd2gndJXjt97eY4jLBsfLgfcOMkZ7d6ZB4juba3EEEFukShwq4Y7Qzo+OW7FAPpn61auvEY+x2rW1vbm8MEqSXBD74S8shKr8237rdcEjPXPQduUNbkM3hPUrfRf7SkUCIRrKymOQYRsYO4rsPUcBieenBxDe6CbFjHJqNk8yyKjwxs5dN3Q/dwR67ckZ6VFd6t9ttVS5srZrhUWP7X84kIXgZG7aeABnbn8eaadWuDri6qFjFwsqygbfl3DGOD9KNL+QtbGr/AMIhNa6tp1vqdykMF5LsEhimjJIIBUB4w2fmGDjb79az7vR1invBaXsFzHaoZHKBwQN4THzKMnLD296dda/NP9lEFtbWYtJmniEAb77bSSSzEnlRS/26BezzrplkqXEZjngHm7JMsGz9/IOQOhHSlp+Y/wCv6+Qo8PTK032i8tbeOCOKR5JC5GJF3LgKpJPPOB+nNWLHwdqOo+abR4pY0lMKSRrJIkrDHRlQhRyOW2jn2OKd9rtzfm582KCMXIiDLGhUKI1wuBnjimW2reTYCzubK2vYVcyRifeDGSADgoynnA4ORxT0DX+v67lBlKOVYYZTgitG40Oa20+1uZZ4t92A0MASTe4Jxwduw/TdkZpItMtJIUd9bsImZQTG6T5X2OIiM/QmnprklvpclhZ28ECyFDJNG0haQqchsMxUHI6hQfpzSVuoPyFu9Fk0+J52uLS6+zSrHcwxs+YmOflbgZ6EZUn69KTX4LeC8tzaW6W6TWkUxjRmKhmXJxuJP60t3r0l2rhrK0j86US3Plq4+0MP73zcdTwu3r9MR6rq39q+TmxtbZoUWMNAZMlVGADudun50Pb+uwLcrWdlLfSmOFoFIGSZrhIh+bkD8KbdWslnOYpWiZgAcwzJKv8A30pI/WnWd/eafKZLC6ntZGG0tDIUJHpkU26u7m+nM17cS3EpABkmcuxH1NIZoQWVja6NDf6mlxP9pldIooJVjwEAyxYq3dgMY7Hmki0Caa3SQXFvHLNG00NrIzeZIgzyMLt7HgkE46dKjtNXMFh9iubO2vbcSeZGtxvHlsRg7SjKecDIPHAqSPxBPHbRoLW2aaKNoobllYyRIc/KPm2n7xwSCRnr0qnb+v67iVxZ4baTwtbXUdrHDcC6aF5EZz5gCKckFiM5J6AU+Hwxez3k9skkHmQXYtGJYgFyGOQcdPkPP0qFtZDaKum/2daBFbeJgZd+/ABb7+MkAcYx7VcPi+7EnmR2VlE7XAuZGVHzLIFYZOW/2jwMD0xzQ7Xv/XT/AIIa2B/CF0rwhb6xdZlLBxIwVV8syAklRjKg/wBcVBHoGNVtLeW7hktriI3H2iDdjyl3FiAwByNjcEUHxLeGJY/LgwsXlD5T08kw+vXac/X8qZZ620V5YPcxK8NrC1uUXgvGxbd174dv0odr6C1tqWdK06x1P+0Ll7dkjhZPKhF9HbgBieskgIPT8aZY29nLFrEL2cZNvA8sMpmLvGQ6gDcpCMME87ee1UJLgWsd5ZWsiT28zqfNKkEhScHB6deam0zWBpkM8Y0+1uftCGN2nMmdhwdvyuo6gH196Wlvkx9fmPh0Cea1jk+020c00TTQ2rs3mSIM5IwNo6HAJBOOnTL/APhGbs6GNVEiG2wrOTFKu0FtudxQK2Ceikmmx+IJo7dEFramaKNoobkqxkiRs/KPm2n7xAJBIz16Us3iOebT5LU2lqrSwR28k6q29kQqV/iwPujoBmnoGo7W9HtNLFu1tqMNyZIY5DEBJuO5ckjKAbfTnNV9YtIYJYLizUpa3kImjQnPl8lWXPfDKQPbFNutTN9aW8E9tbiSFFjF0N+8oOgI3beB/s54p2sXcNxLb29mS1tZwiGNyMF+SzNjtlmJHtik7a27/gCM6iiikMKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAqW2tpby6jt7ZDJLKwRFHcmoq2vCd+LDxFbO7W6RM2HeeNGCjB6Fh8v1GKatfUTvbQy7u0ksp/KmaFmxnMMySr/30pI/Woa6fw9PBcTXl3fGx8zMa+Q6W0ClecsN8bLxgZCjJzUrpao2ojQjpZmF/IB9raEr5H8Gwy/KRnOcc9O1FtAuct5MghExjfyi20PtO0t1xn15pldLbX80/g9rSBtOEkE7l0njt1YxlBypcZZsg8jLdPap9Qk0xfDirZ2drKhgjxN9qgEqScbiY9glJzuGCxXByOMYfLr9wXOTorr7q10yCaeXzdNaOfVonhSOSNitv85OQPuryMg46cjpWDq2o/bJWiW3s444pX8t7eBYzt6AEr94cdTk+9J6f16f5gncpxW0s8czxLuWBPMkOQNq7gufzYfnUVa2h3iWUOqSN5LObQLGkyhgzebGfunhiME45HHIIrUubew1CS6W2k06GRms5cs8cSgeUfN29vvEZVeeOnFNRuF9/wCu3+ZytTx2VxLYzXiR5t4XVJH3D5S2cDHXsa7cXOiQS3Iij0lgZp2QtHG3HnxhMZ7bC5A6EeoFZL2sc+h6okbaWAtzutMzW6S7Az7sEkORjbgHr2pW0uFznYbWa4hnliTclugeU5A2qWC59+SOlLPZXFtb2888e2O5QvE24HcASCeOnIPWupnltofD19HazactrLYwiFEMf2iSTfGX3H7/AAd3BIHoMDinq6NBYaXer/ZRuIkYTrA1s+TuO3dGuQ3y45IPvTaSYk7q/wDXQ56WGSCVop42jkU4ZHUgg+4NMrtL25trzxdHc6g+nS2Eu97cxGBSWKfL5hAJHzY/1gIHcYzWH4kkhkvovKs4LZhH8/kXEMoc5PJ8lVQHHGAB0yetJqw07mPRW54Z8vzrhZIrUl1VRNcSW/7nnqEm+V/oMH3Ga0bBLKIKIJdLnRbyQX0l0sal4cjaY1bkAjdxH82fwp8ornODTbltMbUFWNrdWCMRMhZSc4ymdwzg84qrXQ2lor+G9VEM9oqzTRtCkl3EjsqF8/KzA9x257Zrn1xuG7pnmkUJSojSSKkal3YgKqjJJ9BWzqlzoctjt0y28ufcPm+zunHfk3Dj/wAd/GmeGriK31c+dKsDSQSxRTs2BFIyFVYnsMnr2zmi2or6FC8sLzT5Fjv7Se1dhuVZ4yhI9cGnWOm3OpNIlmsbPGpcq0yISACTgMRu4B4Ga1dMsRZS3STvpzXxhDWhluIZYc7huydxTdtzgN/PFXrKO3bxYJoZrCFY7Yi4ZbiOKPzWiZTsyQCN393j04p2C5yVFdTomn2ZewW9m01PsuoMbszTxkNHiPAzk71yGHGQM8461oWVxoy6fBFJHpZJihDs6Rl8sJd5yec8J9OOhOaLaXC5w1TiyuDbxTCJikzmOLHV2GMgDqeoroLq6sJNNu7VYrBRHp9s8LpGgkab93v+fqTy2RnseM5NQ3TiPXNCdmVLVYLYxtnCgcbz/wB978/jT5Ve3n/X5Cvpf+un+ZmS6LqVvPFFeWctm0xIjN2vkK2OvzPgfrVe8tJrC8ltbpNk0TbXUMGwfqODV6/kj1TVjHZ2lnakyP8AOs7KsnOclpHKj8Mda3b2OD/hJNVmSbTZriZd9k8s8MkR+YBs5JQNtzgP/PFStUmN6Oxx1PWGRoXlWNzGhAdwpwpPTJ7Zwa7GzGmjUrqYtp7OscKywhrdIy5B8xkMiMuAR0Qc544ApkV6kf8Ab2naU+mKn2oPai4SDY6B2zh5BhuCMZPrinYLnKG0nFkt2Yz5DSGMODkBgM4Ppwe/Xn0qGtjTc/2Lrfn48ny48YxjzvMG3Hb7vmfhmsepGFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABU1vdPa7jEE3Nj5mXJA7jnse//AOuoaKA3Lv8Aak4+6kK44TEY+Re6jPY98+/qcn9qTj7qQrjhMRj5F7qM9j3z7+pzSopWRPKi7/ak4+6kK44TEY+Re6jPY98+/qcn9qTj7qQrjhMRj5F7qM9j3z7+pzSoosg5UXf7UnH3UhXHCYjHyL3UZ7Hvn39Tl0esTxNlIbU44QSWySBF9AHBH9ffk5oUUWsHKjRk1u6lAzFZow6NFZxRnHcfKoyD7/1OY/7UnH3UhXHCYjHyL3UZ7Hvn39TmlRRZD5UXf7UnH3UhXHCYjHyL3UZ7Hvn39Tk/tScfdSFccJiMfIvdRnse+ff1OaVFFkLlRd/tScfdSFccJiMfIvdRnse+ff1OT+1Jx91IVxwmIx8i91Gex759/U5pUUWQcqLv9qTj7qQrjhMRj5F7qM9j3z7+py+PWrmHPlx2m3oqyWkUoQeg3qcCs+insHKixeXst9IrzLApUYAgt0iH5IAD9ar0UUFBRRRQAUUUUAFFFFABRRRQAVK11M9ols77oY2LIpA+UnGcHrzgcVFRQAUUUUAFFFFAExu52s1tDIfIVzIEAwCxGMn1OPXpUNFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAFqWx8m1WWa4hR3QOsJ3Fyp6HgYHryamvIoWsrCWKFIGmDB9pYg4bGeST+VRS3/n2qRTW8LyIgjWf5g4UdBwdp9ORUx1c7LYRWdtEbVg0TLvJBznuxByaw9/R+Zv7lmvIfP4fvILi2iJUm5cohKOnIxnhlB79gaij0qSbUTZwzRyOqkswST5cdRgruz7AUT6r9oWJGs7ZYo5Gk8td4DFsZyd2ew70k2qG4uI5J7WGRYoxGkRaTCgHI53bu/rUr21tf0Kl7G+n6lSVFjlZUkWVR0dAcH8wD+lbek6TZ3OiveXMbSOJzEF/tCG1AAUHP7xTuPPasi9vJL+7e4mwHbAOM9hgckknp1JJq1EyS6Sls91FFGJTI25TvDkYH1XAzntz7Z6It8uu5zTspabE9l4bu9Q0ibULVlaOFXZkMUvAUZPz7Nmcc43frVeXSfs1pHLdX1tDLKiyLbt5hk2N0bhSo45wTnHbpWpb6r9lsI7Yvp7NDBJbeeyOzosm7IGG2kfMfmxkc9eMwXNzDd2UcN41g00Maw/agJPMUD7oAVtrADjdtPA78ZpuPT+v60I5u4upeHYoLqyt9Nv4bue6jh2wgSBmZx1BKABc+pz60l74R1CxeMSlNsiSsrNHLH/AKtC7DDordBwcY9+tP8A7QwLN/OskurHYq3Shy/y/dXG7ay9iQvbvxkF5bQ6hHdwR6bbMgdJo4xKyybwVKkFzlcE8rjHPtkbjr8/+ACkU4NBnnETNcW8KSWrXW+RmwqByhzgE5yDwM/nxVu20K0jhluLu5S8i8yOGH7LcLEsjMpbJkkX5AAOQVzn0p9zqXn4Hm2FvGlobPy4UchAW3DHJLcnlueh6nBMFhNFYwTW80tldW0hUywzByFkH3WUoQ3cgkHHXg8ZOaN/67/5Bzf1/XmLJ4cYTTme5ttMjS4NuEu5i5DYBxujQgjB+9gD6VDH4enkgVjc2qTyRtLFbM7b5UXOWBAK/wAJxlgTjjtmfUbw6kJRdXdqgkuDNJ5asdshAGBycrgdfY9eMyxal5NrHG01i08ETQJdFHaSINn5VwdrD5j8xUkc4/hyrr+vT/MOYzW0a4Rbgl4v9HtorluTysmzAHHX5xn8auXHhS8imaCC5tbudLhbaSKB2JjdiQuSVAwcHoTjvirD6oG06S1d9PQyQx289wqOZGCFdgHzYIG0AkAdD1+XNrVNZi/ta4k0q5s7WP7Z57SxK7NK4J2E7icjkk4AHXg/LmrxFzaf12/zM2+8PvoMkEurxST20xZR5Je3bcMf89Y89xztIPrUs/h5L3xDdaZokNwgs2cSyTuZshW25Cxx7upHADdfQGqOqvby7JLeCygZWIlW1aTljz/GzZHuuB+hpF1ycale3csEEy3xbz4HDeW+W3diGGCAQQc8Uk0yvQsP4V1Fb2ezTy5LqEI3kDcHdGx8wDAcDIyDgjuODiS38P2cmmX88us2ivbSpGrKJSnOck4jJOccY9DntVS21yWxmlm022t7OSRlIaPeSgHJUbmPBI5znPTpxRDrPlSXf/EvtHt7shntT5gjUg5BXDBhjJ796NB6jItFuJjDskiImtpblTk/cj357dfkOPwp82iG3sVnlv7NZGgWdbcu3mMhx0+XbnnpnPFSW3iKa2shAlnaMywyW6TOrF0jfdlR82OrHBxn3xxVC6vZLt4WkCgwxJEu0dQowM570O3T+t/+AGtzRvvDl3pWnwalNsmtndRgxTRg5G4D51XIIB5Un69KdqenRT63YW2nwR2xvYYCI1Ziqu4GeWJOMn1pt/4kl1CORJbGzQTXC3E+wODK4BHJLcA7jwMe2KbceIHnvrK7jsLW2lsinlmIyHITG1TuduBj609L/MWtvkWbLTdN1DVNQSKNzbWkG6P/AEpIDKQ6puZ5AVXO4nH0FK/hZrnXrix02RgsaJIvmRSykhgD1iRsgE43YAPUdaqHUbaw1G7exgiu7S7j2mG4DjYCyvtJUqcgqBkHBxTk8SThp/Ns7SaKUxlYnVtsXlghNuGBOAejE575oVtLjd9bf1sIPDlykN7LdTw2yWUxglZ1kYbx2yitj6nANSWnhPUrzR/7RiUeWUaRVMcnzKucncFKDoeGYHjpyMtt/Es9tqdxqKWdsb2eV5fO3SgoW6gAOARz0YH3zVY6t5mnJa3VlbXBiQpDO+8PGpJOBtYA4JJGQevpxS0sGtxXtIJvD6XtuhSW3l8m4G4kNuBZH9ujA/Qepqtc2n2e3tpftEEv2hC+yJ9zR8kYcdjxnHpVmS7gh8PpZW7l5biUTXDbSAu0EIg9erE9uR6VWubv7Rb20X2eCL7OhTfEm1pOScue55xn0oe/9f1uCK9FFFIYUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFPWGRoXlWNzGhAdwpwpPTJ7ZwaZXSaBfyjQr+xgbT1nLRyRi7jgAcAnd80gwSMjAJ45x3prUTObqW4tpbWRUnXYzIsgGQflZQynj2Irq9Mj00eG3W8exkaS3nYlpLdGjkAbYgGzzScgEEMF5x7FuoS28mkyNcHTmhOn26QGNo2uDOFjBzgl1AAYEHC8dMnltWX9eYJ6/wBeRzd7ptzp6wvcrHsnXfG0cySBhnB5Unv2qrXUyWltHb+HxqU9o9tAxW6SK7jkZVaUt0RicYPJHT61ZubiwTVLdptP05RHFcNv+028yS4iOxSsKqB82MZG4k9eKGgTONorrrSSxury0unNgs/9nlpIQtvEssvmsAp3qY0bbg/dyQPfNaBl0K3vLoxJpbq7AjeI3H/HqWIHAAHmgdAOeABnFHLZMV7nA0V3Ucujq1xJb2lhdSNPmWNrm2gUoUUgL5iHjO//AFZUg/hjhnILsVG0Z4Gc4pNWdhp3QlFFS2rQpeRNdLuhDguu0tkd+Ayk/mPqKQyKnxRPPMkUQ3O7BVGcZJrT1i40iaOIaTb+UwJ3nyHjyP8AgU0mfyFZSsyOHRirKchgcEGgOhLFZ3M8jpBBJKyfeEaFsflUTo0bskilWU4KsMEGrtnPHFZXXmpHMWKYSRiM8nngg1d0cWjLuuPs4SSbDo7Rr5a8c/OCxHOOCMY61Dk0S3YxKUoyqrMpCt90kcH6VogAaWn2T7LnD+f5uzfnPGN3PTH3e+afPcNcaPbgG22xgrKNsauPmyMDr09PenzDuZVFb1y9kLi3CwW/2cXCESCWNvkzyCoUN067s9Khjht7YWiztayP5kpcB1YY2jaCR79KXOLmMmOKSaQRwo0jt0VRkn8KWaGW3k8ueJ4nH8LqQfyNaNrd/aEnjYW8MzxbEYKIlf5gSpIwBkcZ49KfM1tarO0Edr5myLC5EqqxzuxknP60czuF9TIorYtJIZY0lCWabpSblZQownGNoPOOv3ef0pIYbVo0lD24jW3mUq7KGL5fbx1JwVwfajmsFzIp0cbyyBIkZ3Y4CqMk/hWzcLFHZkTJapEbaIpsCeaXIUk8fMO/XApmrNbeRttoIQm8eXIk0ZO3njaoDen3snijmBO5mz2d1bKGubaaEE4BkjK5/Ooa1Ee2fV4TOUaMW6AZYbd4iAAJOQOfUEetSu9ub0+dBbxwiE+btljctzwV2AAN06D68Zo5mFzHVGfdsUttGWwM4HrSVtw3AhvbqGBrMJJDiElY9p6YBLd8DnPemWUMMssDXL2imO6Yz7mQArhcYA4IyD04o5g5jHoq+10kFjAsUVu7MHEm6NWbqcc9R7YxVuVbUaK6loHkEKMjhowxbIyMBd3AJHJ5/kcw762Mue1ltljMoXEgypV1YEfgTTZoJLdlWVdpZQ45ByCMg8VdvYSmm2f7yFjGrBgkyMRliRwDnpWezs5BdixAAGTnAHQU07gtRKKKKoYUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABVqz1Caw3+Qlu2/GfPto5sY9N6nH4VVq3p2nPqEkoE0UEcMZllll3bUXIHRQSeSBwD1pq4ix/wkF5/zx07/AMFlv/8AG6jfWbqRtzLbhugKW6IFHdQFAAB78evqc25fDTwz3CTajZJHbpG8kxMm3EgyuBs3E4x27/XDYvDksjhTf2SCSYw27NIxW5YY+4QpwORy20c/XBZsXulT+1Jx91IVxwmIx8i91Gex759/U5P7UnH3UhXHCYjHyL3UZ7Hvn39TmRNDuneJC0atJby3ABJ4WMuGB46/uz+lTS+HJ47Nplu7WWVbdLlrdGYyCN9uD93H8Q4zn2xzS5Q5Y3sVf7UnH3UhXHCYjHyL3UZ7Hvn39Tk/tScfdSFccJiMfIvdRnse+ff1Ob2reFNR0ax+1XajYHEbjy5F2MQeMsoDdDypYe/Iqlo1jFqWsWtpcXC28csiqWOeckDAwDye2Rj1oUbuwrRtcT+1Jx91IVxwmIx8i91Gex759/U5P7UnH3UhXHCYjHyL3UZ7Hvn39Tm9H4akvNXmstMuortYlZ3kjjlbywGxggR7ickfdU9fTNI/hXUVvZ7NPLkuoQjeQNwd0bHzAMBwMjIOCO44OEo3Vx8sb2KX9qTj7qQrjhMRj5F7qM9j3z7+pzMuvXSKFSDT1UcAHToGIH1ZCT9SSat2/h+zk0y/nl1m0V7aVI1ZRKU5zknEZJzjjHoc9qoxaLcTGHZJERNbS3KnJ+5Hvz26/IcfhVWtsCSHPrt3JG0bQ6eAwIJXTrdT+BCZH1rNrTm0Q29is8t/ZrI0Czrbl28xkOOny7c89M54qa+8OXelafBqU2ya2d1GDFNGDkbgPnVcggHlSfr0os+o010MaitTX4LeC8tzaW6W6TWkUxjRmKhmXJxuJP61Ts7KW+lMcLQKQMkzXCRD83IH4UW1sPpcLe9ktoysSx5zneUBIHcfQ/56nMv9qTj7qQrjhMRj5F7qM9j3z7+pzBdWslnOYpWiZgAcwzJKv/fSkj9a0ILKxtdGhv8AU0uJ/tMrpFFBKseAgGWLFW7sBjHY80rJktIrf2pOPupCuOExGPkXuoz2PfPv6nJ/ak4+6kK44TEY+Re6jPY98+/qc2ItAmmt0kFxbxyzRtNDayM3mSIM8jC7ex4JBOOnSlnhtpPC1tdR2scNwLpoXkRnPmAIpyQWIzknoBRy/wBfOwWiVv7UnH3UhXHCYjHyL3UZ7Hvn39Tk/tScfdSFccJiMfIvdRnse+ff1ObkPhi9nvJ7ZJIPMguxaMSxALkMcg46fIefpUz+ELpXhC31i6zKWDiRgqr5ZkBJKjGVB/rijl0uK0TN/tScfdSFccJiMfIvdRnse+ff1OT+1Jx91IVxwmIx8i91Gex759/U5tx6BjVbS3lu4ZLa4iNx9og3Y8pdxYgMAcjY3BFSaZYWOqSXkqwSM6sPs+nwXCxyOpzkhnDbiMD5QMnPHTFHKFolD+1Jx91IVxwmIx8i91Gex759/U5P7UnH3UhXHCYjHyL3UZ7Hvn39TnQtdCtbjS7+5uL5LGW3mWMQ3AkJjyW4fbGcnjt6HIFVo9EL2Ed1JqFlAJlcwxyu4aTZkEfdwOnG4jOaOVD5YkLavcsc4iBUBUPlg7F6bRnt9ff1OY5r+SaIxtHEF6LtT7g67QfT9evqc27ixhj8NWt1H5EkklwyPKkj7l+VTsKlQOM5yCc5qe58K3dvL5Ud1aXMoultHSF2ykjZ2gkqBzg9Ccd8Ucuv9f11FaO5h0Vpalo39nWcNyL61ukmkeNfIEgIK43Z3IvqKzaCwoqaG386N3MiRohAJfPf6A+lSfYWUuJZYogj7MsTyfwBqXOKdjRU5NXSKtFWVhX7HM52M6sBwxyvP0wc/WntpsiwmQyRcLnaCc9A3p6Gl7SPUfsZvYp0VafT5UhSXIKOQAdrDr06gZ/DNBsT5nlrPC75IKqTkEDp0o9pHuHsZ9irRT3iMccbMR+8BIHcDOKntbb7Rbz7Vy6bSCTgKM8k03JJXJjCUpcq3/plWircGmy3O4wsrKG2hgrEMfwHH44pgs5CU5UBgxyTwu3rn8qXtI9yvYztexXoq82k3EdqLl5LQxbQxVb6Evj/AHAxbPtjNS3FtZNBafZIbhJboZUyzqyr85XkBBnpTlLlaTIjHmTd9jMoq++kss3lR3dtLIN4dUZsoVUsc5Az0PIyKW10W6u2hEW0iWIy5AZtqhivIUE9R2Bqfawte5Xsp3tYz6K2bLSFXW/7K1BMSXKBIJASux25Q4ODgnAIIzgnuKx2UqxVhgg4I9KtNNJrqQ04tp9BKKKKYgooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAroPCV5b2E97NPdpaSGDZC7mQAsWBwTFhwMA8qfr1wefopp2E1c6XV57OS4u0tL2Ax3Plm5fMj5mUclCxLFSSeTk8H2Bisb9bG2ihkksJ/s0heJ5UdjBIccrtIDDIBOQRweP73P0UtVsLlb6nSwauYLXynk0+SVYZbc3Do7SKJNxIGGwRlidwGRz1GAZL/UrcwiKyms4WNrBbz3KiQysVRflHzFdu5fvKBwvf8AiwbTTLq9t5Z4REIoSA7yzpEMkEgAsRk8Hgc0R6ZdSac18BEtuCVDSTohYjGQqkgtjI6A9afvCtruad/Jb6rJunk0y1uGbdcXS+dl378AsCD1yqgZ9sZpPBHprxXVnq1pczRSKyrCkoYEHIPzxqO3rVWeyuLa3t5549sdyheJtwO4AkE8dOQetQqpZgq8knAoV09Nx201NT+3iLyaZNOs0juEKT248wxy5bdk5fIOQD8pHSm22uS2M0s2m21vZySMpDR7yUA5Kjcx4JHOc56dOKpXlnPp95LaXaeXPC210yDg/UcVDST7FWNOHWfKku/+JfaPb3ZDPanzBGpByCuGDDGT371JbeIprayECWdozLDJbpM6sXSN92VHzY6scHGffHFZawyNC8qxuY0IDuFOFJ6ZPbODTKfSwFi6vZLt4WkCgwxJEu0dQowM571o3/iSXUI5ElsbNBNcLcT7A4MrgEcktwDuPAx7YrGqeeyuLa3t5549sdyheJtwO4AkE8dOQetF2Itarq39q+TmxtbZoUWMNAZMlVGADudun51Ws7+80+UyWF1PayMNpaGQoSPTIqvRS63H5E11d3N9OZr24luJSADJM5diPqat2mrmCw+xXNnbXtuJPMjW43jy2IwdpRlPOBkHjgVnVat9NubuznuYFjaO3XdIDMgYDjnaTuI5HIFMC3H4gnjto0FrbNNFG0UNyysZIkOflHzbT944JBIz16U1tZDaKum/2daBFbeJgZd+/ABb7+MkAcYx7VTjsriWxmvEjzbwuqSPuHyls4GOvY1BRdi0N8+L7sSeZHZWUTtcC5kZUfMsgVhk5b/aPAwPTHNQHxLeGJY/LgwsXlD5T08kw+vXac/X8qx6kkglinMMsbRyg7SjjaQfcGjV6BZLU0bPW2ivLB7mJXhtYWtyi8F42LbuvfDt+lQxXNpYXEyLZ2upR7/3cs/mrx9Fdf1zVa8s59PvJbS7Ty54W2umQcH6jioaLvcLGouv3DXF7LeQQXgvmDzRzBgpYHII2MpGMnv3qnPevcWtvAyoqW4YJtBz8zZ5yfeq9FIZptrKtoq6b/ZtoEVt4lBl378AFvv4yQBxjHtVqw1/frIk1EJHb3GoR3lw0aElSpboDnj5jxyawqVEMkiouAWIA3MAPxJ4FVd3uJpWsdH4h1Cy1EWrzXa3FysjB5bdriVVi4wMTtktnPAwKxNQuIrrUJpraBYInbKxqMAD6dqL6wuNNuBDdqiuUDjZIrqVIyCGUkH86rqpZgq8knAqOW8rlc3u2J7eeOK2mV0WQsykK+ccZ54I9abLcyTKwfB3PvJx3xj8qLyzn0+8ltLtPLnhba6ZBwfqOKhpcqvcv2kuXlJ4roRW7xeRG4f7zMWyfToae1/KylSqYIx0P90L/IVVopckb3BVZpWTLUmoPID+6jUs4dioOWI/Go4m33YkeUQnduLYJxznjFQ0UKCSsgdSUneWpZlZLu6kcyJAvARWDdOgHAPahZms8i3mjk3YJZQ3GPqBUUFvNdTrDaxSTSucLHGpZm+gHWlubS4spzBeQS28q9Y5UKsPwNHIrW6B7R3b69x/2sEMrW8TIW3BDuAU98YOacJli09olcO8rZIAPyDuOe5wPyot9NubuznuYFjaO3XdIDMgYDjnaTuI5HIFVaHBAqsty6+tarJZ/ZJNTvHttoTyWuGKbR0G3OMVF9sk/wBF4X/RvucdfmLc/iar0VTSe5CdtEWIb2SG/N2FRmJYsrA7SGBBHrjBNWW1mRmQfZrcQrCYfICtsZd27nnOc85zmq4028Z7dEtpHkuV3QxoNzuMkZCjnsanTQtSbUEsZLR7a5kGUjuiIC3OBjeRnJqXSjJ7FKrJdSTSr21tdV/tGVEiNsPMgt4wxDyD7oyScAH5jk9sCsxmLMWY5JOSfWlkRopGjcYZSVYehFNq1siHuFFP8mQQiYxv5RbaH2naW64z680+W0ngghmljIinBaN85DYOD+IPbr09aAIaKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAK1fD4tzd3HnG3E/2dvsv2rb5ZlyMbt3y9N2N3GcZrKoprQTOntQ2Lw/8AEn/tXzY/9b9m8nytvOz/AJZZztzjn9av2smiILtrG0s7v/S33xy3MMAMWBtCmZCdud33CrDv2xxNFO4WOgGsFfBstqgs1Y3YVYzbxM4Qo3OSuSe27r70mpau1x4V022zZlt0okVLaIOgBXbyFyucdRjPfNYFFK4WOovra2aPRZ9QbThGrBL4WcsG7b5ndYjknZ3A/Wq+oC8/taAJ/Yu0SP5Hkm18vb/t9sYxjzOfxzXP0U76hY7LyNNt/FmoyOLCVJIy9kkdzbiIncMjJDRqdu7AYfkcGq01/aWzX8sNjp8U5nt1WNzDcgDa+9lwNnPGcDAz0BxjlqKFK1gsdjHfIh17T9KbTFX7WHtROlvsdA7Zw8gw2ARgE9M4rj2zvOcZzzjGP0pK0Ytcu4YUjSKwKooUF9OgY/iSmSfc0ugzp7+40dtMuYYY9MDbJAjIke/iKIjB653l+fqOgxVSS2gNx4fnvDpOElVbwQy2+CPN43Kh5G3qcfWuSJyxJ7+gxRTcru5PLpY6yzutLvG06bUIrGOQTXKmOJY4l4RfKD8EY3E/MwPvkZpLi6s49SnlNjYQyR2LFAZoLhJJNw2nEahNwHYDtznJzylFJvQdtf67nU6FdW93PdXl4mnpIBEnkFbeFWHO6QeYjKMYGVQAnNOt7SA6l4hFjPZR20iSwWwe8iQN+8UqBuYZGB16e9cpRTv+v4hb+vQ6G0tTN4TvI5G00SrKjQFprdZsAtv5yHI6cHr2p9ykEnhWPmytHjRMIpt5ZLgk8nK/vUOCSQcjjHHArm6KVwsdlrr6fa6Sr2hsJbq3ukEMifZnMse1ssY0QYBIHD7iPbvX1+4ub3xElzavpUkUrl7dv9FGQVB/eZ/9qd65Winza3Cx19xbWkPiDVHtW0syyLvsN0sDQD5hu4yUB25wGx+eKjnvLGzW6kt4tMkujJaq/wC5SRFba/mlFIK4yADgbeeOxrlKKFKwrF7Wlt116/Fl5f2YXD+V5Zyu3ccYPpiugtEt4dBsXuodNW0ksp2nZxF9okffIE2g/PkELgjj1OBxgW2sXNpbrDFFZMq5wZbGGRvxZkJP51Bd3096YzcFCI1KoqRqiqCxYgBQB1Y0tFGw93c6Wb+zRZvv/s/+zDbx+R5fl/avN+XdnH7zP387vl6Y7VFrKWz2s8SnSwGuEGnG1aIN5fOd5XkDG3Pmc5/GuXopt3BKxueJoNklk4mtZVW0iiPkXMcpDKgBBCscfXpWVZ3cllP5kKwsxGMTQJKv5OCPxotLyWymMkKwsxXbiaBJR+TgjPvV+PxJfQyeZHFYLIOA66fACB3HCY5/z1NF9bhrax0ITTbbXdaX7Nam4WdPJhkkghQR4O7b5qMnXbxgH0PWq2nW+m3l/bSsunWcMOpyPcRTXEZAiITaoJ/1i5DDjI57A5rBm1u7uLiSaXy2ldi28oCQDyRz2PfPv6nLP7UnH3UhXHCYjHyL3UZ7Hvn39TmeZ9ibM27W6sBY21o8Wn7H0+5aWRo08zzQZfLG88g8LgDBOR14p+oyaavhtUsrO1lQ28eJvtUCypJxuJTYJSc7hgsVwcjjGMH+1Jx91IVxwmIx8i91Gex759/U5a+pTPGyBY0BBVdqYKKeqg+h/Pr6nLcm/wCvUet72IbVoUvImul3QhwXXaWyO/AZSfzH1FaOsXGkTRxDSbfymBO8+Q8eR/wKaTP5CsmijpYrrc2dAkjNvqdoJo7e5urbZDJLII1OGDMhY8DcARzx271a0y1+yW1zFG2ltqaypg3U0EkflEHO1mJQnOM98dO9c5RTuKx1Vpb20t9rrafPZxW00LwwCS6SLcxZThQ7A44OCaZY2li1lBcTS6eqw6fcxyo8qb2mzJs+XqTgrhunA5zgVzFFK+ny/wA/8wO4vrjRm024hhj0sHY4RkSPfxFERg9c7y/P1HQYrN166sLm21JYIrCPyNQVbX7NGiExYfJyvLDIU5OQM8Y6VzNFNyu7glZHUzTpb/EKWW4MYt13+VvbCGLyj5YzkcFdoqhpEkd54msZooLPT4oZUdx55RAFYEnMrnn2B7dKyprqa4jhSZ96wpsjyBkLnOM9ccmoqIys0+wmrqx1MECwSagLd9LfUDcBla6mgkjMJyTtLEpnO3P8WOnetHSE0P7dcMXsXs5b0oySvDGIo8D5h5iGQgkkAKRjGcjqOFopJ2VhtXOmt7+Wfwg1pbNpweC4csk8durGMoMFS4yzZB5GW6e1UI/+RNn87p9uT7P9djeZj8PLz+FZFTTXc88MMMshMcClY0xgLk5PA7k9+tDd7/L+vwGQ0UUUgCiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKtWGnyahM6pJHEkUZklllJCxqOMnAJ6kDABPNVat6dqMmnTSMkcU0c0ZililBKyKcHBwQRyAcgg8U0JliLRRL50n9pWaWkTKn2p/MCO5GQoGzdng/wAOOOvTN2z8F6teNMqIqmKYwD5JHDuMcbkVgByOWIHPXg4pxa0IhNF/Ztm9rK6yfZXMpRHAxuB37s8nqxHPTpTm8QTXCSLqNpa3weVpl84OvlswAO3Yy8HA4ORxT0/r+vUNS7Z6ZYW3hq4v7qWxnuPOESxTm4HlnaxKjYAC5IHUleOtNuNP0+x8KQXAlsbq5uWfDn7QHXG0YQYVcjJJ3fhmscX0g0trDanlNMJs4O7cFIx9MGia+kn0+2tHVBHbFyhAOTuIJz+VK+4WL9/paiHSY7BIZpbtCBJDI58595UZDqu0jpxxxmq82mwW15FC+qWcmWKysglKxY9Ts59MruqebxA7w2aW9haWr2LBoJYjIWQ7t38TkHnnkVBcapDcXiXB0myQhmaREMoWUn1G/j/gOKel7hqX5vDs994ovrDToEhW3JYqhkmWNcgcYUu3JH8OfwBpj+FbqC4uY726tbQW7RKzzlxu8wErhdu7t0IBHcdcMn8SST6hPdNp9mBdJsuIR5mybkEE5fIIIBG0jpVN9TPkzQwW0FvDNLHKY495ClAwABZicfMc5JoXLpcNdTSi8NwLY6k19qVvbXNlcLCVYSFVOWBztQ5zt4x75xWARhiAc+471qjXpWur+W5tLa5S/k8yaGTeEDbiQRtYEYye/emR6baSxrI2tafCWGTGyXBKexxGRx9TSAuXPhG7trGS5e7s28tSzRq7FuFViPu4ztdT17468Uy50dJP7Ft7I25e9Gzz0kkKyMZCuWDKCuOnAPTNJceJ7yeOWJo7crIHBKq38SIhxz6Rj9aRvEbYsPJ0yyhawkDwMhlJGG3YOXIIJ/H0xTfLfTbQWtvMVfC93LcWsVncW939paRA8BdghjALgjbk4Bz8oOe2adP4TvbW9NvczQwr9nNx5sqyRrsBwflZA4Oe23ntniqtrrtzaRQRxpCyQvK211JEgkUK6tz0IXtg89aYdUCyTta2VvarPAYXSMyEYJBz8zE54HfHtSdrD6/13/yLVn4bl1C/ktbO8t59iLIJIY5pVYH2SMsMdDuAp9np8cSa1Y39pE1zZwuwl3vuR1dVwMNtI5PUGqenau+n280H2W3uYpXSQrOG+VlztI2kf3jwcj2qzF4jdLq/uJtOs7iS/ZjMZPNHDEMVG1xgZH196elvk/8AgBrf7iOCyhbwxdXf7mWZJo1z5jh4Qd38O3awOOucjHvUd3o0llplveXFxEpuU3xQ7ZNzLnqG27D7/NkU631pbfS57EaZZuk5DO7mXdkZ29HA4ye31zQutvFpEmn29pbwpNt851aQmQqcgkM5XOR1Cj8iaNA1LF14WubKyt7y4uIktZnVPOeGdAm4Eg/NGCwwDyob+VGsaLZ6brrWcWp27xCQqd4lzEB/fIj7/wCzmo9S8RT6nbTQvaWsHnzrcTPCrbncAjJyxx948DA9MVFd6yt7eR3V1p1o8o5lOZR55xjLYfj1+XbzRpcWtizfaLv17UYYvs9jbWjZkZpXeOIZAAzt3NkkAfLmmr4am3TGW/soYYjFid3fbIJAShUBSTnHpkdxwcLP4lkuL64uX06y23S7biEeZsl5yCfnyCCB90iq1zrdxcxyRtHCkbvEyoikCMRhgqjnphj1yT60Ll0uPUqXlrJY309pPjzYJGjfacjIODWjb+HZ7iyhnS7tRJcQvNDblm8x1QsG/hwPunqRntSSxRavcTahd6tYWk1zI0jwsk+VJJP8MbDH4mrU+sw2NhZ21gtrcXENtJC14Fk3JukkyFDEDlW6lcjd2PRfZ13DroVX8O3KwsRPbvcpGsr2as3morYwem0/eXgMSM9OtJd6LJp8TztcWl19mlWO5hjZ8xMc/K3Az0IypP16UsniK5eE7YLeO5eNYpLtVbzXVcYBydo+6vIAJx165S716S7Vw1laR+dKJbny1cfaGH975uOp4Xb1+mG7dBK/UTX4LeC8tzaW6W6TWkUxjRmKhmXJxuJP61BpWlzaveG3t2VSqNIzMrNgDrwoLH8Aat3N9Frs0S3CafpfkxBFlxOQyqAFXq/bvge5qHba6XMkqT2OrZyPLH2hAh4wc4jP5E0aXfzHrZFjS9Es73+0fO1W3QWsJkRwsu1uVG7/AFZO3nHQHOOMUaX4YudZeddOnjmMT7AVhmKv6HcI8KD/ALRWo28Q3Mmr3F/PDDMbmPypYX37HTAGCd27+EHO7ORyTUlj4ll0/YINPsmENwbiAOJD5LEAHHz8jCj72TRp+AtSQ6AtzpthJazW8dxJayStA7tvmKSSZI4IGFUdSM44yc1Xngt5fC1rcxWkcVx9qaBnRnJkARSCQWIzk9gKIvEE0NnHClpaiaKJ4Y7rD+YiOWJA+bb/ABsM7c80w60P7HTT1060RUfzFmVpfMD4ALffxnAHbHtQ7a/11/yH1LGreFNR0ax+1XajYHEbjy5F2MQeMsoDdDypYe/IrJtrZ7u4WGJolZuhllWNf++mIA/OrWpaoNTJklsbWK4dt8txFvDSHuSCxUZPPAFVba6uLK4Weznlt5l+7JE5Vh+IpaXDWw+8sZbGRUmeBiwyPIuI5h+JRiBVywsbX+yp9S1ETSQxyrAkUDhGd2BOSxDYAC+hzntVO81C91GRX1C7nunUYVp5WcgegyamsNVeyt5reS3gu7acqzwT7tu5ejAqQQeSOD0JoQMsW2iG/CzQzw2cVxKY7WO6kJaUjHAKpjuBk7Rk/XDoreBvDV+0tpGl1aTRqJgz7juLZBG7b2HQU238QPboqrYWbrFKZrYOHP2Zjj7vzcjIBw24cfWmwa35Om3Fm+n2k/2lg0sshl3swJIPDgcbj2+uaNLaBqJHoN1JNJGskOY4IZzlj92QoFHTr+8Gfxq/L4MvY2iUXllI0kqxBUd+CZDHk/L0DAj+Warp4nuEtTEtnZiRo4onn2NvdYypQH5sD7oHAGe9O/4Sy+81JPKt8pIJB8rdRMZfX+8xH0/Oq90XvWIZPD0yzWUcN1a3IvJzbo8LMQrgqCDkD+8ORkHsas2WnadqGt3cMSP9ltoGaP8A0lYTKVIG5ncFVzkn0HSqFvrFxbrZKix4srk3MZwcljt4Pt8g/WnXF5Da3142mENb3cRXa6nMasQSv1BGM9DS0t9/5D6/13/ysXtOtLFvEU1jLZRyQvE5Xfd+cYysZbIkiKq3I9D6VSsdDlvoIpWura28+QxW6zswMzDGQMKQOSBlsDnr1qPSdV/si5adLO3uZCpVTOX+UEEHG1l6g981Yg8QPbqqrYWbLFKZbYOHP2Zjj7vzcjIBw24cfWjTqDv0HWvhi+vNLnvYSu23DmRDFLxs5b59mzOOcbv1pdQ0W0tNFsryLVIJJZ42dotsmWIcjC/IAMd8nqDiki8TXMdosLWtrJIsMsC3Dq28JJu3DhtucsecZqo2qGTSY7Ga1gl8nd5M7bw8YJyQMMFIznqD1NLSwD7+0g/s2yv7RCiTBopU3E7ZUxkjPYhlP1JrOrR1C7g/s2y0+0cyJCGllk2kBpXxnAPYBVH1BrOoe4LYKKKKQwooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooqe4s5rVInmCbZRlGSRXBH4E0rpOw7Nq5BRRRTEFFFWbPTr3UWddPs7i6KDLCCJn2j3wOKAK1FBBViGGCOCD2ooAKKtyaZdQ6fHeyiJIZRlA06B2GSMhM7sZB5xilutKu7OOFrhYlMwBVBOhcAjIygO5cgjqBTswuU6KmvLOfT7yW0u08ueFtrpkHB+o4pbOyuNQuPItI/Mk2M+3cBwoJJ59gaQEFFFFABRUhgmVY2MThZf8AVkqcPzjj154qa80y/wBO2f2hY3Nr5mdnnwsm7HXGRz1oAq0UVPaWN3qExisLWa6kC7ikMZcgeuB25oAgop80MtvM8M8bxSodrI6kMp9CD0plABRU8dlcS2M14kebeF1SR9w+UtnAx17GkhtZriGeWJNyW6B5TkDapYLn35I6UAQ0UVNa2lzeziCyt5biVgSI4kLsfwFAENFS3NrcWdw0F5BJBMuN0cqFWHfkHmltLK6v5vJsbaa5lxnZDGXbHrgUAQ0U+aGW3meGeN4pUO1kdSGU+hB6VONNuW0xtQVY2t1YIxEyFlJzjKZ3DODzigCrRT3hkjjSR43VJASjFSAwBwcHvzTKACiilVWdwqKWZjgADJJoASir0ei6nPcz29tYXFxLbttlW3jMmw+h259D+VUmUoxVgVYHBBHINACUUUUAFFTXNpPaMi3EZQyRrInOQykZBBHWoaACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooA1boAaZF9j+xmDyV83Pl+dv/i+98/X04x+NSzWw+xaWklzbKEYrKRNHJsDPnJUE54/+vWLRWPs3pqbe0XbpY6O+Nn51i8cNq8qzOHjE8IDKANu4qoQd+oPv6VCwij1pGxa3G6EkpugjWI8jGeY2I+nPoO2FRUKhZWuXKvd3sWtTRU1CQRzRzDg740VFzgZGF+XjpwSK1IoxqXhW2tLO4t4pre5keeKe5SHfuC7XBcgHG0j1GfesGiuiOkbHPJ80ro6q0FimmQZ/s37H9mkF35pjNx53zY2/8tOuzG35fXvTr+bTovDyCxsrSVfIiKzG5t/MSTjcTHs8087hgsVwc9MY5OircibHRa5rcky6WYvsLmO1RmCWkJ2vlvlIC9MH7vTvina5qMuoeKU+yNp77Anlu0UCxk+WudzEBW6EfNnHQVzdFHNrcLaHWXFtZReKtQaNtMIkid7AebE0G/IxkAlF43YDYGcVDo6SSa/crqJ0cRtEwm3NaiPJjOzYR8oOcZKfjXM0UrhY63RFsLPTXSa1sry7S4YXCSXtsgKYG3a0isCPvcxsD+mIrW70/wCw21q0NgqSafcvKzohkEuZTGN55BGFxjBOR14rl6Kbd0FjqpdRlvPD+mbH00Jb5S4Ux28cq/vMgqCA2MEcr15z3q5O9hbX9+uq3dtPZ3+qJKkcE6y4jDsWkOwnblTj+9yeOK4mijm1v/X9aC5dLHW381r/AGtp+NMsGKSOSVv7QpKuOAxRFRcdRvBznHPSnNJp9rfakbu4hmim04BorUwQlW81P3YaMFGYAZyByOoHbkKmt7p7XcYgm5sfMy5IHcc9j3//AF0r6DaOz87TZNV1CdTaXsrrAbXzJ4Ix5OzBBMyuu8YUEHDcEjjIqlpv2Sd9QQQWGnxyTtieSa2n8hQPuhJOXH+0mD6bsAVgf2pOPupCuOExGPkXuoz2PfPv6nJ/ak4+6kK44TEY+Re6jPY98+/qcjk2TZmnZ232jwjdq7aaJBIjW5eWBJsAtv5JD+nB69q0ZpbWDw/fR202nLayWMAhRDH9okk3xl9xHz8HdwePQYHHN/2pOPupCuOExGPkXuoz2PfPv6nJ/ak4+6kK44TEY+Re6jPY98+/qcnM10DW9ze1K1sIY7yWJtNm8/U43too548+T85wSD8inKgg4xjkDirTi3OqBluLCBrizuYxbo9sqxEphQZYsI2SeN2CMdO55f8AtScfdSFccJiMfIvdRnse+ff1OYri8e4jCOkYVT8m1cbB/dHt357/AFOS/T+trAr3OrtJdPjFpZ30lreXtpZOEJni8sOZAwTzHVoyQhbqCuTgHNQXLwX1nqdjZLaafcvNDK0ZvYvLmVVIOJBtjzuYNtGB1wOK5Oim5X3KSsdRodhZs1gt7NpqfZdQc3Zmnjw0eEx3+dchhxkDPYHNV7C0L+F9TAns1aaSJo0e7iRmCF93ylge47c9s1z9FK4WOlvr+W+8KWIjbTtsCPHOhjt0lU78jaMB+QRyvXnPerV1a6ZBNPL5umtHPq0TwpHJGxW3+cnIH3V5GQcdOR0rkKKrm1uLl0sdLZ6jZXniBIr+2sUtopJTC0cSRAnaQis2CCuQOXDe+RmrKSRSeLrP7PZ29tOIJPL8m4hlDy7H8s/uVVAd20YAHQE9a5GlVmRwyEqynIIOCDUp6JMbW50Ph51/su9jMVjPL9pgdY7268kYAkywO9MkZHc9elbWltpM+p3c91cWt1Bcag4ka5e3RljOPn+eMuQ2T9zbjGeO3D3FxJdXEk87BpJGLOwUDJPfAqOnzf193+QmrnSWaW8nhiZGNlalVkfzma3lkmPZNp/eoeMBlyPbqan1GTTV8NqllZ2sqG3jxN9qgWVJONxKbBKTncMFiuDkcYxylFJvSw+tzXvf+RV0vzv9b5s/levlZX9N+/H41DrX/H5F/wAeP/HvH/x4fc+6Ov8At/3vfNVLm7nu3RriQuUQRoMYCqBgAAcAVDQ3cEFFFFIYUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRVqztIbrf5+oW9ntxjz1kO76bEb9cUAVa2dO0a0vdDu72fU4LaSGSNQsgkIUNn721D1xxg+uah/sqz/6GDTv+/dz/wDGqs2clvp0dxAbqxv7acKJVdJgu4HKkY2Nxk5P14PGXotyWyOz8NXd/pM2oWrq8cKuzL5UvRRk/Ps2dOcFs/jxU2o6Cqwmeymt/wB3ZQXElsHYyAMibn6Y+83TOfbGKng1b7NYpb7tOLRQSWwmKOzosm7KrhsEfMfmxkYPXjLJ9SWWzaHfYQSNBHbT3EYkMjqoAVeWK4+VcsoHQ+2RuNtP63/4AlLX+vIgvbCO8g0X+zbKOG4vlZCkbsQ7CQqD87HHT1xUs/g3Ube6t4pJIkWcSFZZVkiUbF3NkSIp6d8Y9+tSyaqnl2QgGnWklg37qWHzWZG3bsfM7BlJPJx6+2YWurdLrz7ddOs3EckUoh81gTIpXoznIAJ5GAOeDxkbiCkQp4bnk1C3tYbqCYXMJmhlhjmdXUEg/KqF85U/w9qsL4Mv/tE8MtxawtC20+azru/d+ZwNuR8oJ5AxjBwaW31L7PGkU0tjNDHamzkjdXIYGTzByrAn5j95SBx0PGZ316ZpHZp7BM/K6pG2FPkmEBQDyu09R0I+gJeNmHMQW/gjVLqSZYTG4jk8pXWOVlkbAPBVDtGGHL7Rz7HHPMpRyrDDKcEV0cmqJcIyaj/Z10gfcFcS/unKhflKOCQQoyeRwcDpnOXSrUqN+uafG3dCs52n0ysRB/AkUaN6DUr7mZT4ommmSNCoZyAC7hR+JOAPqavvplokbMuu6e5AJCKlxlvYZiA/WqEUskEySwSNHIhDK6MQVPqCOlHXUfQsXmmz2Kq072rBjgeRdxTH8QjHH41BbtEtzG1wpeIOC6jqVzyKnvNV1DUVVdQv7q6VDlRPMzhT7ZNVKQ+hchggNvNdzrIYlkCJGjAEk5PUg4wB6URWH2nDpJHAkjlYVmYkufTIGO45OBUdvdmCOSJoo5opMFo5M4yOhGCCD1796kj1JolAFvCwRy8W4MfKJ9Oeeg4OajUnUkj0W7ltDcBcABmAKtyFznkDaOh6kfyp72ttBpEcnmW8sszN8zeblcYHGABxk5z+FVmvjJbiOeCKVlBCSNuDKCc9iAeSeoNRPcNJbRQkDbEWKkdTnGf5UWk9w1L17bW1taW6Rvbu8ihmlHmbuSeeQBjj0zUdzZbr6KC2WMb4lbcrkqflyWywBHrjtVaS5aYw+YikQoEA55AOeeferD6o7TwzR28ETxLsG3cQy4xtIJPGOKLNBqJbWsLalDEbiCZGYZ5kVW5+7nbkE+uKW20ua9mnEI2rE2CdrvjrgfKCex5xUcd6kV2k8dnApTBCZfbkHOfvZz+OKd/aPzS5tYGilIZojv2hh3B3ZB5Peh36BqSJo07PseWGJ/OMKo7HLOMcDAPr1posIf7LW5N3GshkK7CG7AHHC9fxxUcV/JC8JjjjUQzGZF5xk4469PlFNjuylq8Dwxyozbhv3ZQ4xkYI/XPSj3g1Lk1ha/vYoPOWaKFZtzuGV8gEjAAI6+p6Uy50S7tY1aQDlwhG1lwx6DJAB+oJFRy6pJJGwWGKJ3VUeRAdzBQMDknHQdMdKZcXouXEkltCJC253XcC575G7HPsBQuYFcsQ6WBqq2czmVtrlkhDKwIUkD5l749DTn0hpbxYbVZEPlCSSOUFmi5xg7Rk9j070kOoGWWNWMNrHCjqgw7D5htPUk98+g54PQziZQEE91ayKkXkSBgx3c5UcHkD+8MYx34zPvCbaZU/sedZ5o5XSMQoHd2V8AHocBSR+IGO9MtbeOS3u2JjkaOMlQWYEcj5hxg/Q461bhFtb3JmEloCpHAaT923baQ2SPU5OMH2y+KSKJ7h5pbOR5siYsH4JOcDaRlc9x0wfbLu7BzFay05XQyXLxjdBJJHES25sKcNwMdR3Paq7WMirKSyfu4klPJ6Ntx+PzCtFJkihCG4s8pG0JfazMobPyjnBHJ+btg9eMo0qG1eF7izUtGsMsiqxY7cbQOcEcDJA7Hrxku7hzalePStmoW1vcyrmZl3IgYMAe+SuD9RmhdOS9SJ9PWRSzmNkkbeRgbsggDIx2xnjvmrD3f2WCP7M9oBbyCQxKXOX6cZY5GCckY6fTNP+05FaPyYYYo03fulBKtuGDnJJORx1+mKPeYK7Jm0hrVpHvd3lRxCT5AVZ8ttA+YAjnuR2qo0KyxvNbqUiV1TbI+5ssD3AH9005L7ypmaK2hWN02PD8xVx75bPXHQ9qWO/wDKMmy1g8tyreWdxVWXOCMtnuepIpq49SWK2SM39tcRRvJAjESBmyCGA45xj6imixhOlLdG7jWQuV2kN2AOOF6/jihNUKzXMr2sEj3JO8tvHBOSBhhxn8ahjvNls8DQRyIzblDbvkOMZGCP1z0o94NRWsZFWUlk/dxJKeT0bbj8fmFSyaXIkohWeCScuqeUrHdk9OoA/I8UjanI1q8PkwgvGsbSAHcVXGO+Ow6CoWvJTffa1IWUMGGBwCKfvBrYlvtMn09Y3mwVckA7WXkezAHv1qC6aF7qRrVDHCW+RWPIFLcTpOwKW0UHXPllufzJ/SoaFfqMKKKKoYUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFavh8W5u7jzjbif7O32X7Vt8sy5GN275em7G7jOM01qJmVVq3025u7Oe5gWNo7dd0gMyBgOOdpO4jkcgVsRWovFvYryXS4rwXED7hJBHGIwrhtpT5T1XIXk+hIq1DbQyar4jaznsIreVZobcG6iiVv3ikBQWHGB16U7aff+X9IL6/cct5MghExjfyi20PtO0t1xn15pldNb300vhBrKFtNEtvcOXSZLcMYygGVLDLHIPK5bp7Us9rYDS727WWwxJp9usEayIZPNBjEhCDlTw2ScZyevNJr+vkCf9fOxzFFdLdajp8niJLOe3sRpa3MZMkEChgg+9868kHJzyfbGKi8TyQMIFis7WFwzfvbe6t5d68YBEKKB65Iyc+1DVlcE7nP0Vt+GovMbUmSG0lmjtN0X2zZsRvNjG7L/AC5wT1+nOcVevLuxtTqEtrHpz3Qjt14hjePzMHzTGpBUjPoMdx2NFguc7bWVxdx3ElvHvW2j82U7gNq5Azz15I6VBXSaT/xMrbVmlTSoFnjYxJI0ERWXK42byGUYz0O2n2lxYpY21lNHp+G0+5M0jIhfzQZNg39QeFxgjOR1GKGrfd/mF/zOYqeeyuLa3t5549sdyheJtwO4AkE8dOQetbGohRosH9mnTjZm3j83Pk/aPN/j+9+8+9/d4x7ZrRFkk974cacaXI/nJHdR28lud2ZeAyRnn5cZOPrT5dbC5tLnH1YsbGfUrtba0EZlf7qySrHuPoCxAz7V1AS0e6sE1BtLgulmmYi28gx+UEGxGbDR5LAgFskZye1FzFZyeLNHms2sYNixy3WLqAIjLKc5ZQiE7QOFHPv1pxim1cJNq5yMiNFI0bjDKSrD0Iptddpsdna3ep/aFs7m8MqtCGu7bYYyWzh5FeMnO3jhvTuK5rUWjfU7loIFt4zI22JJBIqDPQMOCPcVHRXK6srUVt2l1oSaSI7q23XexgX+zO3POPmFwo9P4PwNY8DIlxG0q741cFl9RnkU7a2FfS5NLpt9DZpeTWVxHbSY2TvEwRs9MNjBqKCB7mdIYzGGc4BkkVF/FmIA/E10d9b203i3+0r+6tbnTLm6D7luVZtjcqGQHeoAwCMDGMelRapB9pgtYZTpa6i0z4NpJDHH5WBjcyEIOc4yc+vamkFzDvLSawvJbW6TZNE211DBsH6jg1DXWanaw3PiDWR9osHN1DutZDdRFdwkQ/e3YU7Q3Ug/nV+N9FtJJUA0qbARdxVGVsWhJxn1kA56547mi2l2F9ThKlhtZp4ppIkykC7pGyAFGcDr7npXYreaVFqVrGsWltDcX5FwzRRkLEY4s47IuS/Ixgg4I5rAfP8Awh6+Rjb9vbz8dfuDy8+3+sx+NDVl/Xl/mCd/69f8iidNvlsBfNZXAtD0uDE3lnnH3sY68VDJDJCVE0bxllDKGUjKnoR7GuiuhHdaQbvVfskd5bwRrbS293HKZsbQqSRAtjCg8/KBjkE1Y1G9/tCbSJ5W0xrPFuswVLdJFYcMGUAOF4PH3cYp8utvMV9LnJqpZgq8knAqW8s59PvJbS7Ty54W2umQcH6jitSbUJbzVEhtbfTI/Kmcwv5EMabeysWwrDA4LZPvWjffZNO1vWbiJdMlRoN1mqtFLGG8xBwoJAIG44I6Z4xUrVXKejscpRXdWcNhNc313GtkilbT955Nu0as0e6RQsrLGCWHY5GDgYzXN6jZTXetaofIhszBvmNuNqhFB4VdoweCOnBpTtDcI3k7IyaKK0pQiWvzrAqGBSuNu8vgc+v9KiUuVo1p0+e/kZtPkieIqJBjcoYc9QelWLmYGUxRJDs+XaQgGDj1/wAasOsaTrK/2Y/uDkKUI37T2HvUuo1bQtUU20nsZlFa7m2e6BTyAF34UbAG6YGSCO55NQzvBFLO8IhJ2Jt4VgDxnHGP0pKrfoVLDqN/e/qxnUVft2iuFd5Y41MLeaQq4DL/AHfzx+Zqix3MSQBk5wBitIyu7GMocqTvuK8bR7d4xuUMOeoptaMzRPZoImj81Yl37yCSPRc9D+tOupYI1UwwRMiuCh3oSR6FQM8++azVR9jV0UtbmZUtvaXN2xW0t5Z2UZIjQsQPwrQhNhaXsK3kYltzmRv3Zc4I+UYDofQ/eHXvUstzpba7YSaZGIIUlQyExNGPvDn5pZO3fI+lW5Nw5kZOPLPlZkz209rJ5d1DJC+M7ZEKnH0NR1o2k8cEF7JtgeXK+V5qK/8AFyQDweKt6etteyW80/2KDbeF51cqg2YXgA9RkNwOmewrN1HFXaL9mm7JmHU0tpPBBDNLGRFOC0b5yGwcH8Qe3Xp61swvZJpH7u3gnYrJ5u+eJGDZOMBl3njaRtI/nmvH/wAibP53T7cn2f67G8zH4eXn8KuE+e+m3+diKkOS2t7mRRRRWhmFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUVb07Tn1KaVEmhgWGIyvJMSFCgjPQE9+mKAKlFbNn4bl1C/ktbO8t59iLIJIY5pVYH2SMsMdDuAqaHw5CNP1Fr/UYLW6sp1hZHEhCnLA52oc5xxg+uadmK5gUVqxWMB8M3d0PImmjmjXcJJA8QO7+HbtYNjrnIx05om0CeG1kka5tjPFEs0toGbzI0OME8bf4gcBiRnp1osMyqntLyWymMkKwsxXbiaBJR+TgjPvRZWj39/b2kJVZJ5FjUscAFjgZ9ua0G8Oy+ZbCG8trhLiZoBJAJHCSAAlSAm49RyoIPY0JPoLQZ/wkN95ciotpH5i7GMNnFGSuQSp2qMgkDr6e5zB/ak4+6kK44TEY+Re6jPY98+/qc35/Cl5a6gbW5mghAtjcmWUSIuwHBO1lDA5zxtye2eMusfD9pcNfCfWLTFvbedHJGZCrZK8n92TgZwRgNntily33FZMzv7UnH3UhXHCYjHyL3UZ7Hvn39Tk/tScfdSFccJiMfIvdRnse+ff1Obdhp9vJaaqxktrp7eEmPDyoRhl/eL8mCOcYYjr0p9p4T1K80f8AtGJR5ZRpFUxyfMq5ydwUoOh4ZgeOnIycoWiUf7UnH3UhXHCYjHyL3UZ7Hvn39Tl8OtXdtOk1t5UUkTAxMsY/dgc4Ge31z39TnPrYn8NzxXC2sd7Zz3jSpF9midt4LdDyoBHTOCcZ5xQo9gcY9SodVnzlUhU9FxGPkXuo9j3z7+pyf2pOPupCuOExGPkXuoz2PfPv6nOhd+ENStLi2ibaRclgrskkQXaMsSJFU4A5yAR+NVY9GE00gh1Kzkt4YxJNdASiOPJwAQUDEk46Kev1wcqCyIf7UnH3UhXHCYjHyL3UZ7Hvn39TmrNL50pfaqZ6KgwAK3NM0uKDXDYahBb3iTW7SRzK74x5bOrLgr1wOGH4CodO0a0vdDu72fU4LaSGSNQsgkIUNn721D1xxg+uaOUNEY1Fblr4S1K70k6hCAYijyIPLk+dVzk7guwdDwzA8dORnDAyQKdmnYoKKu3Wk3FnB5s0lmy5xiG9hlb/AL5Vif0p2j6emo3zJM7JBFE88rIAW2IpJAz3OMfjQBQorTjsbbUpZH09WsbWCPdPLeT+YFycD7kYPJIGApNXNM0uKDXDYahBb3iTW7SRzK74x5bOrLgr1wOGH4CiwrmBUsN1NBFNHE+EnXbIuAQwzkdfcdatWWj3F99j8p4x9suTbR7ieGAXk8dPnH61oW/hC8uLNLgXdmm9FdY2dt2G3bei4ydjDr2545os7XC6MCitefw9NBp73Ju7V2jgjuHgRmMixvt2k/Lj+IcZz7Y5pX062juNKsGBM9z5ck8oblVkxtUDpwpBz6n2p8rvYLq1zHorb1a0s9Kv4hDaLIiuwZJdRjuBIBwM+TtKfif5GnaloiN4i1KG0MVnZ2jZaSVmKRKcADgMxyTjgE1K1HsZ9lq13p8MkNu0TRSMGaOaBJVJGcHDggHk8j1qKe/u7l5mnuZHNw++UFjh29SOladr4Xu7vUZLSKeJmWNZUeOKaVZEbow2IxA5/iAqSHw5CNP1Fr/UYLW6sp1hZHEhCnLA52oc5xxg+uabTe4k7bGBTndpCC5yQAo+g4FaFvaQXei3jRri6syJS4JxJESFPB6EEqfoxz0FZtKw7hRRRQAUUUUAP85xCYgQEJyQABn6nqaet3IihQsOAMcwoT+ZFQ0VPKn0KU5LZgTk5pUYowYYJHqAR+RpKKom/UdJI0sjPIdzMckmm0UUDbbd2FFFFAgqaa7nnhhhlkJjgUrGmMBcnJ4Hcnv1qGigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAqe1vJLRZxGFPnxGJtw6AkHj34qCigDQ07V30+3mg+y29zFK6SFZw3ysudpG0j+8eDke1SjXpWur+W5tba5S/k8yaGQOE3biQRtYEYye/esqnrDI0LyrG5jQgO4U4Unpk9s4NO7FY0LbWVttLnsf7Ms5EnILyOZdxIzt6OBxk9vrmpG12a7jMMkVpDLOiwTXpV97RjA+bkjsMlVycd+c58dlcS2M14kebeF1SR9w+UtnAx17GoKLh5m9pVrp+n6zZXk2u2TxwXCSMsUU5YgMD3jA/X8+lWF1ZbeeE25063ggMga3RZGR3kXaSSWLMCvGQRjBx2zzNFF2Kx0VxqPnYXzbG3jS1Nn5cKOQgL7uMkluTy2T0PXjNXTnj02aR/tNrMkkZhnikDFWz2+UgkZAO4HjHfjOPT44ZJQ5ijdxGu5yqk7V9T6DkUtd7hyvub1le2th9qAh0x1uAY5QxnOwEg7FxIMrkD5uTweTxljXMElglrd/2fL5KtHHM3m74ckkKu1gGUEk5IOOevGci2sri7juJLePettH5sp3AbVyBnnryR0qCjXqFn3NM6VZhSRr2nk4+6I7jJ9uYsVO9wx1kan9vt4rlJA5KKSFcdMDncvqfY9eM4tPkhkhKiaN4yyhlDKRlT0I9jRrugaubovYINQivLQadZyRFhIsSyOrlhgqQzNlcEglcY5/2ckd5bQTS+XHpsdvLGIri1HnNG7ZyDkuWODj5lYYweufmxruyuLCRI7uPy2eNZVG4HKsMg8e1QUarQLN9TpbfVVt9SN7J/Z0jrH5KoyybYBtKhUCsNwwcFjnoTycEwW1zBam6RvsD21wVEts3mlEcHK7SH3EDJ53Hv14zg0Ua9w5X3N1rmCTT0tbv7BMYVaJJm8zfDkkhV2sAygknJBxz14zhAkEEcEdDRRRruNKxduta1S+g8m91K8uYs58uadnXPrgmo9Pv5tNvUubfaWUFWRxlXUjDKw7ggkVWooGaketLDLJ5OmWaW80flzW2ZSkmDkEkuWBBA6EfzqS38RSQakb1rGzlcR+VEjiQLCm0rtUK47HGTk9+vNY9FO7FY2bDxJJpzo0GnWR8qc3EAkWRhCxAB2/PyMKPvZxjjBpYfFF7DFGixW5EaxqMq38G/Hf/pof0rPt9NubuznuYFjaO3XdIDMgYDjnaTuI5HIFQeTIIRMY38ottD7TtLdcZ9eaG3sw0Lr6zcOtwpSLFxbRWz4B4WPZgjnr8gz+NS3OoxyHTr+EqLy2VI5ImU4by8bGz6EAAjrx71lUUXYWROl0PtxuZ7eKcM5ZonLBWz/ukH9a05/EslxfXFy+nWW26XbcQjzNkvOQT8+QQQPukVi1JBbzXU6w2sUk0rnCxxqWZvoB1pLTRDfdmqviWcGYS2dnLFJ5e2F0bbF5YITbhgTgE/eJz3zUY16Vrq/lubW2uUv5PMmhkDhN24kEbWBGMnv3rPubS4spzBeQS28q9Y5UKsPwNS2+m3N3Zz3MCxtHbrukBmQMBxztJ3EcjkCnd7isixb3cFpot2sbZur0iIqAcRRAhjyepJC/gDnrWbRRSGFFFWRpt4z26JbSPJcruhjQbncZIyFHPY0AVqK0E0LUm1BLGS0e2uZBlI7oiAtzgY3kZyaoyI0UjRuMMpKsPQigBtFFP8mQQiYxv5RbaH2naW64z680AMoqaW0ngghmljIinBaN85DYOD+IPbr09ahoAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAK09Emgt5LyWdLd2S1YwrcIrLvyuMA8E9eKzKKAOp0K6t7ue6vLxNPSQCJPIK28KsOd0g8xGUYwMqgBOasRXqR/29p2lPpip9qD2ouEg2Ogds4eQYbgjGT64rjqKq4rHQ2tsbjwpeLK2mCUSo8G6W3SXALb+chsdMA9e1WLz7CNLn8v+zvsJtU+y7PL+0+d8ud2P3nXfnd8uOnauWp0chilSRQpZGDAMoYceoPBHsaV9LBbW5e0CNJvEemxTIskb3USsjjIYFhkEeldAEtHurBNQbS4LpZpmItvIMflBBsRmw0eSwIBbJGcntWJH4k1CKVJIRZRSIwZXi0+BGBHuEBqudVnzlUhU9FxGPkXuo9j3z7+pyc1lZCaf9fM6TUJtMgv/ALRDFp/m/wBmMzR7oZUE/mED7iqhbGOi4Pv1NTQ9aklub9Nmm28tzaFE328KRtINvXcNq5APHCk9s1jf2pOPupCuOExGPkXuoz2PfPv6nJ/ak4+6kK44TEY+Re6jPY98+/qcrmf9fP8AzFr2/rQ2NLge4g1eK7OkgvGwQvJaoRNuXGw5GBjP3flqe3fTYfDKKlna3UnkyC5L3dvG6yZbBAZTIcDaRsbBxjHXOB/ak4+6kK44TEY+Re6jPY98+/qcn9qTj7iQrjhMRj5F7qM9j3z7+pyXdrBqU1Yo4YYyDkZGR+RrptcvZ9Rt9OmgfTGiMUKkBLZHWQDBDDAYLkd/lxjtWWdfvCpUw6fgjHGm24P5hM1D/ak4+6kK44TEY+Re6jPY98+/qcu+lh63udN/Z8M3iixlddOkjNpulW1eB0WVYiWYop2gBufmwpxycVM/2JNUjNza2ZxZH7LIBaQx3Em/5iSoeIELkYbPTsSK5i3128tJfMthDGwBRf3KsFQjBTByCD3znv6nK3PiC9umUyC3UJkRrHAqLGpOSqqBgA9/X8Tl83b+txWYzXXik1eUwW0VsuFBSKZJVzgZIZAF564UADpitXQLqxgt9NhuYrBhPfuty9xGjMsO2Pu33Ry3PHQ4PWsuPXLqJdscVns/hWSyil2j0BdSQPx9T1NU7m5e7nM0qxKx6iKJY1/75UAfpSTsVq9zprX7AumwY/s37D9mk+1+b5ZuPO+bG3P7z+5jb8vr3qO6urCTTbu1WKwUR6fbPC6RoJGm/d7/AJ+pPLZGex4zk1zFOjkMUqSKFLIwYBlDDj1B4I9jTuFtbk+m+V/atp9o2eV56b9+Nu3cM5z2xW5Z6jZXniBIr+2sUtopJTC0cSRAnaQis2CCuQOXDe+RmsxtfvWUr5dimepj0+BDj0yqA1F/ak4+6kK44TEY+Re6jPY98+/qcq7Wwnc6CR7GXWXzaafa3K2beRvuYJYZJd3Bby1WIHbkAEAZAz1qG1DYvD/xJ/7V82P/AFv2byfK287P+WWc7c45/WsX+1Jx91IVxwmIx8i91Gex759/U5P7UnH3UhXHCYjHyL3UZ7Hvn39Tk5n/AF/X9IWpu2lvbS32utp89nFbTQvDAJLpItzFlOFDsDjg4JqO3vppfCDWULaaJbe4cukyW4YxlAMqWGWOQeVy3T2rG/tScfdSFccJiMfIvdRnse+ff1OT+1Jx91IVxwmIx8i91Gex759/U5V3a3l+tx63ub2oyaavhtUsrO1lQ28eJvtUCypJxuJTYJSc7hgsVwcjjGOZtWhS8ia6XdCHBddpbI78BlJ/MfUVM+pTPGyBY0BBVdqYKKeqg+h/Pr6nNSne7uCvaxraxcaRNHENJt/KYE7z5Dx5H/AppM/kKfoEkZt9TtBNHb3N1bbIZJZBGpwwZkLHgbgCOeO3esaijqM6PTLX7JbXMUbaW2prKmDdTQSR+UQc7WYlCc4z3x071PaW9tLfa62nz2cVtNC8MAkuki3MWU4UOwOODgmuVoouFjp7G0sWsoLiaXT1WHT7mOVHlTe02ZNny9ScFcN04HOcCtG+uNGbTbiGGPSwdjhGRI9/EURGD1zvL8/UdBiuHoquYLa3Om166sLm21JYIrCPyNQVbX7NGiExYfJyvLDIU5OQM8Y6U+adLf4hSy3BjFuu/wAre2EMXlHyxnI4K7RXLVLNdTXEcKTPvWFNkeQMhc5xnrjk1Nwt0NXSJI7zxNYzRQWenxQyo7jzyiAKwJOZXPPsD26VfggWCTUBbvpb6gbgMrXU0EkZhOSdpYlM525/ix071y1FO+39dv8AILHdaQmh/brhi9i9nLelGSV4YxFHgfMPMQyEEkgBSMYzkdRl29/LP4Qa0tm04PBcOWSeO3VjGUGCpcZZsg8jLdPauZopX0sCVnc14/8AkTZ/O6fbk+z/AF2N5mPw8vP4VkVNNdzzwwwyyExwKVjTGAuTk8DuT361DQ9Xca2CiiikAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABVvTtOfUppUSaGBYYjK8kxIUKCM9AT36YqpU9reSWiziMKfPiMTbh0BIPHvxQBpWfhuXUL+S1s7y3n2IsgkhjmlVgfZIywx0O4CpofDkI0/UWv9RgtbqynWFkcSEKcsDnahznHGD65qhp2rvp9vNB9lt7mKV0kKzhvlZc7SNpH948HI9qlGvStdX8tza21yl/J5k0MgcJu3EgjawIxk9+9VoLUIrGA+Gbu6HkTTRzRruEkgeIHd/Dt2sGx1zkY6c0TaBPDaySNc2xniiWaW0DN5kaHGCeNv8AEDgMSM9OtNttZW20uex/syzkScgvI5l3EjO3o4HGT2+uakbXZruMwyRWkMs6LBNelX3tGMD5uSOwyVXJx35ytA1uZ9laPf39vaQlVknkWNSxwAWOBn25rQbw7L5lsIby2uEuJmgEkAkcJIACVICbj1HKgg9jU+lWun6frNleTa7ZPHBcJIyxRTliAwPeMD9fz6VYXVlt54TbnTreCAyBrdFkZHeRdpJJYswK8ZBGMHHbJolqJv8Ar7yrP4UvLXUDa3M0EIFsbkyyiRF2A4J2soYHOeNuT2zxl1j4ftLhr4T6xaYt7bzo5IzIVbJXk/uycDOCMBs9sVJcaj52F82xt40tTZ+XCjkIC+7jJJbk8tk9D14zV0549Nmkf7TazJJGYZ4pAxVs9vlIJGQDuB4x34yrr8/1/wCALm/r7v8AghYafbyWmqsZLa6e3hJjw8qEYZf3i/JgjnGGI69KfaeE9SvNH/tGJR5ZRpFUxyfMq5ydwUoOh4ZgeOnIzNZXtrYfagIdMdbgGOUMZzsBIOxcSDK5A+bk8Hk8ZY1zBJYJa3f9ny+SrRxzN5u+HJJCrtYBlBJOSDjnrxkuvw/r+v8AgBzGDWre6CbFjHJqNk8yyKjwxs5dN3Q/dwR67ckZ6U06VZhSRr2nk4+6I7jJ9uYsVYNyf7cXVWvbVbiOVZCApZQ4xgAd145PsfbL0T1G2OvPCOo2d1aQtsb7UX2OySRBdgyxIkVSAAc5xiodO021fxBaWrXtneRO65wZkSQlseXny9wJ9cY561fbWSslq0L6dapaySMY4kd1ZpAAwO5iWUgYJzxzj+HMEFzZ2uqRXsUGmxmDH7oGdkWQHIYfvMt9c44PB4yRlG6uJy0KtvopvHvJPtNrZQ286xMZncgFi20DCkn7p7e/rixY+EdQv7u7tYWTz7WYwunlysCw4+8qFQOOrEUXF1HKs6rJZWyTTJLKkQkYCRQ2NuWJKnecn247ZvQ+IHhumuJW02WVbt7oM6SHypnxnaA2GXgcnOMHHbKTVtf62/4ISlroZMNjCfDN3dYhknjmjUnzHDwg7uNu3awOOucjHvTZdDaGzWaS/sxK8Kzpbb28xlbGMfLtzz0znirtteW9tpc9i6adIsxHnSN5xfeM7cbXAIG48gY45zxmpc7bmSF5b6BGt4khzGCcFRhcevuw9D7Zd0PmV9CbVvCmo6NY/artRsDiNx5ci7GIPGWUBuh5UsPfkVk21s93cLDE0Ss3QyyrGv8A30xAH51qavcxX0bTPHYLdbt08sBkDyOevBYqQepKgDIPtnLtrq4srhZ7OeW3mX7skTlWH4ilpcad0PvLGWxkVJngYsMjyLiOYfiUYgVcsLG1/sqfUtRE0kMcqwJFA4RndgTksQ2AAvoc57VTvNQvdRkV9Qu57p1GFaeVnIHoMmprDVXsrea3kt4Lu2nKs8E+7buXowKkEHkjg9CaaBli20Q34WaGeGziuJTHax3UhLSkY4BVMdwMnaMn64dFbwN4av2ltI0urSaNRMGfcdxbII3bew6Cm2/iB7dFVbCzdYpTNbBw5+zMcfd+bkZAOG3Dj602DW/J024s30+0n+0sGllkMu9mBJB4cDjce31zRpbQNRI9BupJpI1khzHBDOcsfuyFAo6df3gz+NX5fBl7G0Si8spGklWIKjvwTIY8n5egYEfyzVdPE9wlqYls7MSNHFE8+xt7rGVKA/NgfdA4Az3p3/CWX3mpJ5VvlJBIPlbqJjL6/wB5iPp+dV7ovesQyeHplmso4bq1uReTm3R4WYhXBUEHIH94cjIPY1ZstO07UNbu4Ykf7LbQM0f+krCZSpA3M7gquck+g6VQt9YuLdbJUWPFlcm5jODksdvB9vkH6064vIbW+vG0whre7iK7XU5jViCV+oIxnoaWlvv/ACH1/rv/AJWL2nWli3iKaxlso5IXicrvu/OMZWMtkSRFVbkeh9KpWOhy30EUrXVtbefIYrdZ2YGZhjIGFIHJAy2Bz161HpOq/wBkXLTpZ29zIVKqZy/yggg42svUHvmrEHiB7dVVbCzZYpTLbBw5+zMcfd+bkZAOG3Dj60adQd+g618MX15pc97CV224cyIYpeNnLfPs2Zxzjd+tLqGi2lpotleRapBJLPGztFtkyxDkYX5ABjvk9QcUkXia5jtFha1tZJFhlgW4dW3hJN24cNtzljzjNVG1QyaTHYzWsEvk7vJnbeHjBOSBhgpGc9QeppaWAff2kH9m2V/aIUSYNFKm4nbKmMkZ7EMp+pNZ1aOoXcH9m2Wn2jmRIQ0ssm0gNK+M4B7AKo+oNZ1D3BbBRRRSGFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUVp6JNBbyXks6W7slqxhW4RWXflcYB4J68UAZlPWGRoXlWNzGhAdwpwpPTJ7Zwa6bQrq3u57q8vE09JAIk8grbwqw53SDzEZRjAyqAE5qxFepH/AG9p2lPpip9qD2ouEg2Ogds4eQYbgjGT64qrCucrHZXEtjNeJHm3hdUkfcPlLZwMdexqCuhtbY3HhS8WVtMEolR4N0tukuAW385DY6YB69qsXn2EaXP5f9nfYTap9l2eX9p875c7sfvOu/O75cdO1JrS4X1sctRWhoEaTeI9NimRZI3uolZHGQwLDII9K6AJaPdWCag2lwXSzTMRbeQY/KCDYjNho8lgQC2SM5PamlpcLnH0+OGSUOYo3cRrucqpO1fU+g5FddqE2mQX/wBohi0/zf7MZmj3QyoJ/MIH3FVC2MdFwffqamh61JLc36bNNt5bm0KJvt4UjaQbeu4bVyAeOFJ7ZpW6ev6/5Bf+vuMC2sri7juJLePettH5sp3AbVyBnnryR0qCul0uB7iDV4rs6SC8bBC8lqhE25cbDkYGM/d+Wp7d9Nh8MoqWdrdSeTILkvd28brJlsEBlMhwNpGxsHGMdcjWl/ILnJ0Uqbd6787c8464rsdZkst8KaVZacE+0r9muHurZlCc8OgRWwRjPmk4x1yeWkDdjj443lkWOJGd3IVVUZLE9ABSMpRirAqwOCCOQa66W+03T9W0270j7GLNbrM+6NHkVt3zEBgWEe3Gwjp/vVBNceTrF/PfDSpSLaRrbylt3QneNpwnBbqcNz68Uraff+AX1OXorqdCure7nury8TT0kAiTyCtvCrDndIPMRlGMDKoATmsPWltk12+Wx2G2Fw4i8s5XZuOMHuMUNWsC1KVFbZ1GKz0Sxjgt7GZ5ElFx5kCO/LELk/eUgcggg1qXEemL4RkRns5ZltonhlWSASGQsu5dqoJMgEg72OcZx6OwX1schU93ZXFhIkd3H5bPGsqjcDlWGQePatzW1PnxjTTpZ07zU+ylDb+ZjHHmZ/efXfxnr2q3dR2tvr1tdzjSpIjYNvjikhaMzCBuCkZwPnx6ZPSi2l/62C5yNT2dlcahceRaR+ZJsZ9u4DhQSTz7A109pJY3V5aXTmwWf+zy0kIW3iWWXzWAU71MaNtwfu5IHvmpUjsW8XXQtTpYsHtx5nnSQbQ5h/gLAYO/+4Bj0Ao5QucZRW7BdLo+kECHT57xbwh/MjjuMoFHAPI2k9x+B61ft5tMXwypisbSd2ik+0B7qCJ0ky2MK6mQ4G0jY2DjGM5yrbhc5OilXG4bumea2dUudDlsdumW3lz7h832d0478m4cf+O/jR0uPqYyI0kipGpd2ICqoySfQVPeWF5p8ix39pPauw3Ks8ZQkeuDV/w1cRW+rnzpVgaSCWKKdmwIpGQqrE9hk9e2c1Y0yxFlLdJO+nNfGENaGW4hlhzuG7J3FN23OA388U7CuZVjptzqTSJZrGzxqXKtMiEgAk4DEbuAeBmqtdbZR27eLBNDNYQrHbEXDLcRxR+a0TKdmSARu/u8enFR6Jp9mXsFvZtNT7LqDG7M08ZDR4jwM5O9chhxkDPOOtFv6+Yr2/ryOWorubK40ZdPgikj0skxQh2dIy+WEu85POeE+nHQnNZt1dWEmm3dqsVgoj0+2eF0jQSNN+73/P1J5bIz2PGcmhrcd9bHPiyuDbxTCJikzmOLHV2GMgDqeoqeXRdSt54oryzls2mJEZu18hWx1+Z8D9a07pxHrmhOzKlqsFsY2zhQON5/7735/GqV/JHqmrGOztLO1Jkf51nZVk5zktI5UfhjrRZXt5v8AT0KN5aTWF5La3SbJom2uoYNg/UcGoa7G9jg/wCEk1WZJtNmuJl32TyzwyRH5gGzklA23OA/88U+zGmjUrqYtp7OscKywhrdIy5B8xkMiMuAR0Qc544AoUb2Fc45YZGheVY3MaEB3CnCk9MntnBp5tJxZLdmM+Q0hjDg5AYDOD6cHv159K6uK9SP+3tO0p9MVPtQe1FwkGx0DtnDyDDcEYyfXFY2m5/sXW/Px5Plx4xjHneYNuO33fM/DNLoPqY9FFFIYUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUVqeH9Lt9W1P7PdXa2yeW7DO7LEKTxhT6ZOe2cc09wMuitew8Pvql9LbWN5DP5ahvMiguHVgfZYywx7gCmjw7e/bYbVmiSWa8eyALH5ZF25zgdPmHT3osxXRlU6OQxSpIoUsjBgGUMOPUHgj2NaUOgzz2IvUnt/soDebKWbEDD+F+M5ORjGc59jjLo1TGasfiTUIpUkhFlFIjBleLT4EYEe4QGq51WfOVSFT0XEY+Re6j2PfPv6nMl5o5sZEhnvbb7VvCSW6iRniJ/vEJtOO4Uk1dufCGoWt5aW8jxD7XvEbyJJEAUGTkSIpHGOcY9+tDVydGZ39qTj7qQrjhMRj5F7qM9j3z7+pyf2pOPupCuOExGPkXuoz2PfPv6nOhZ6JbOl4G1CynVbTzkuEdwsREqKdwKhs4J42nORjJqJ/Dc0Lu1ze2kFsoTbdOXMcm9dy7QFLdMnkDGOcUuUVolT+1Jx91IVxwmIx8i91Gex759/U5P7UnH3EhXHCYjHyL3UZ7Hvn39TnQg8KXcsnlSXVnbytctaxxySMTJIApwNqkYO4cnA9ccU218Jald6T/aEagR7HkVTHIdyrnJ3BSg6HhmB49xk5dLjtEqvrt3JE0bQ2ADDaSunW4P4EJkfWs6it210KODXbCzv7m3lkkuoop7NDJvQMRkFtoX2OGJp2bY20kYVFa8nh6UTWixXdtMl3K8SPCJHCuuMqQEyT8w5UMDng1Lc+FL2zvWhu5oYI1g+0NcSrIiqm7aMqUD53cY25/Dmiz3H1sYdX7bWLm0t1hiismVc4MtjDI34syEn86kvdCnsvNLTwSpG0K74y2G81C6kZAOMDnOKvWvgrVLua5ji2EwXDW25UkdXdeoBVCFHI5baOfY4dmK6MxtXuWYsVhDdisKrtHdQAMAH0x6+pyn9qTj7qQrjhMRj5F7qM9j3z7+pzTZSjlWGGU4IrWm8Ozw2bTi7tZJEt0uWt0ZjIsbBcE/Lj+IZGc98Y5qOVWuLlV7FX+1Jx91IVxwmIx8i91Gex759/U5P7UnH3UhXHCYjHyL3UZ7Hvn39Tm/ceE763tbW43o0V1KkKMYpYwGcZX76LkcHlc/yotvD0DavaWlxq1kyy3Hky+SzlkOQCPudTnAIyuR1quTWwvdtcof2pOPupCuOExGPkXuoz2PfPv6nJ/ak4+6kK44TEY+Re6jPY98+/qc37bSbNfFEFkb20u4TMFKkzIr/NjyywjyCfUDHvTNP8OXmr3V4LRRHHbybWISWUKSTgYRWY9DzjHHPUUlG9rdRuMVuU/7UnH3UhXHCYjHyL3UZ7Hvn39Tk/tScfdSFccJiMfIvdRnse+ff1OYb2zm0++mtLlQs0LlHAOcEVNFpNxNafaEkswmCcPewq/H+wWDfhilZNXDlRSop8UTTTJFHyzsFX6k4rXvdMtBqh0fTIbma+SbyTNJMoR2HBwm35RnuWPFVYdzForTm08ab5F2JbPU7ZpCh8tpAm4DlTkI3Qg5HHvU2s6Uket6tHYqkUFl+82FicLuVcDOSeWHU0WGY1Fb0HhC8n3ZurOIBVIMjsM7ovNxwv8Adzntx9MqnhG5eYRG+sUd5zbxBnf96+1WG35ehDjk498cUcr2FdGBRWjHaQw6FLe3C75ZZTbwJuwFIAZ3OOuAVAH+0fSruoaRZ2eiwTpGzTSwRyl21CHgt1Hk7d5H4+9Fh9bGM11M9ols77oY2LIpA+UnGcHrzgcVFW5eael9Bo39m2UcNxeo4ZI3bazByoPzsccDnnFQnQHaW2Fte211HcSmFZYFlYK4GdpXZvPUdFOaLO4royaK6S38ImLXY7DVruO3WS3aZG2SqXADdA0eRgqc7gOBxniqOnWFpPrH9nPPHc/aV8uC5hLqqSn7pwwBIz8pyO5IosFzJqY3c7Wa2hkPkK5kCAYBYjGT6nHr0qJlKsVYYIOCPSkpDCiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAq/o85s75bsTRR+UcYlBYPuBGCBzgjOSOg98VQooE9UdFb6hBaidFj0wQSsjPARK6xyLnaV+fLD5jnJI68HjNiDXngvGupW02WYXb3QZ43byZXxnaA3zLwOTnGOOcZ5WnrDI0LyrG5jQgO4U4Unpk9s4NGvcnl8zeg1D7NZ/ZPOsfsxRkuIijMJXboxwcnGBhhgDacDn5qJ0u0C5GvaeSBnaI7jJ/OLFZlFGpVn3Ojl1NZL6G/c6el9DKsklyiyFpJB0yu7aQepKqOh9suk1bLW/ltp1sls0p2RLIw3yKFPJYlhx1zxg9RgHm0QySKi4BYgDcwA/EngVautMu7O8S1miDTSBWRYnWTeG+7gqSDn2o1Jt5l7Tnh05p/Mls7iOaLyZo5d5GdysMbGBIyo5B7Hrxm4dW8xpVvW0+4t2KD7M6ybIWUbU2bWDEAcE5PfOTjPNspRirAqwOCCOQaSi77j5X3OiGqzC8t7mW9tGltrtrknYcea23jC9U+QcjGOevGYmuYZNPS1vTYTmFWjSdhJviySQF2sAygknJBxz1GM4VFGuwWfc0zpdoFyNe08kDO0R3GT+cWK0JdTSS/gv5P7OW9gmSWW4RZN0ki9MqG2kHqSoHQ+2cePTLqTTmvgIltwSoaSdELEYyFUkFsZHQHrVSm7ha/U6Cy1BrFLdBc2bpbvKWV1ZtxlUKy8HlcD7wxjnrxmY6wCwWT+zfsy232V7UJIUxv3ryG3HDH7wPY9eN3M0Ute4cr7nXXur239oSyJJYXEMkdvvjlibbHNHGFBUKRlQS3qMZ4PGasuqJcGf8AtH+zrpZJ3mKuJf3Mj9dhRgSpwM8kDb9N3N0UO7Dlfc010q1Kjfrmnxt3QrOdp9MrEQfwJFa2oajbGLyLOayjLWsNvcXKrI0jlUUbRklSuV5KqOF75+blqKd30Cx1F1rjXDFmbToXN0lzM0SSEvMucdWOV5OduAMHA6Zy1VY9QF4L+FJkl8wlQSFkzkY9Rnv7HrxnLopa3vcOXS1zolvrWLWE1EQ6YkkbbmjXzigl3ZDD58nnuDt4PB4yf2hAJLkSx6a1vOytLbHziiyjOGUh93c5IbHXg/LnnamtLK6v5vJsbaa5lxnZDGXbHrgUa7BZ9xLpVW6kCeWBnOIiSq+wJySB0zk/U1Yi1rVYLT7LBqd5Fb4K+SlwwTB6jAOO9VZoZbeZ4Z43ilQ7WR1IZT6EHpU4025bTG1BVja3VgjETIWUnOMpncM4POKFsUVlYqwZSQQcgjtWtN4hllvI75bO1iv1kEjXUYfdIw6llLFOe+FGayKKd2KxpzayJvJQabZx2sTtIbZPMCOxGCSd+7sOAwHH1qabxI8+o3F2+m2X+lRmO4iHm7JeQ2T8+QcqOhHSsarqaPqMl19lhsp5rjyxIYokLsqnBBIGSOo6+tGoaGhJ4tvpJHfyLVd/8KIQB+5MOAM8fKfz/KoT4kvDeWtz5cG+2uPtCDacFtqLg89MRj9ay5oJbaZ4biJ4pUOGSRSrKfQg9KZRd7hZbGjHdwzaHLZXDeXJFMbiBtpIYsArofTICkH/AGT61Vu7yS8aIyhR5USxLtHUKMDPvTxpty2mNqCrG1urBGImQspOcZTO4ZwecVVoYzZfxHIEshbWFnavZNmCSLzCV5JIIZyCCSc5FCeJJYJomtbGzt4Yy5NuiuUcuu1iSWLfd44Ix2rGqS3gkuriOCEBpJGCICwGSenJ4ou2KyRpP4hnOoWd1FbW0Is4zFHCitsKEsSDliTncRnOfx5o06/tINX/ALSeCO2+zDzILaEOyvIPu8sSQM/McntgVlyRvFI0cqsjoSrKwwQR1BptFwsKzFmLMcknJPrSUUUhhRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRT/ACj7UeUfagBlbfhg2Qvbg3qRSSiA/Z0mkjRC+4Z+aRWQHbuxuGPocVj+Ufajyj7U07CaudVayW0muXuyxsLYMiLve6tJViOOXUOBG+epC7cdARTNNvWjsdX06zuNNkk89Xhe4hgRJlDMCQZBjuCATxziuY8o+1HlH2ouFhrZ3nOM55xjH6Vv6gFGiwf2cdONn9njM2fJ+0eb/H979797+7xj2zVOLV7yGFI0i08qihQX0+Bj+JKZJ9zWcY2LEnHPpxQnbYOtzpdYsrJbW/a2l05nudRRrVYpo8rCQ/ofkXlcg4xjkDipLmYaX4h0a9M9lLFFFbxSlJYbjZtA3cAtjHY/ka5byj7UeUfampW28vw0FY6W5iN5qWpRXcmlebLbn7I8UluiE+ap+8mFDbQ33iDjjvS3s9jpz6g1kNOnlSK0WJvLSVS2weYVDDB5znjv61zPlH2o8o+1JaKyK6/15f5HWaPdadLYS3D6fYy3UlyzTwPPBbqEwNoXzVbAzu+4VI/LHIuQXYqNozwM5xTvKPtWjFq95DCkaRaeVRQoL6fAx/ElMk+5oeoti9eautx4d0q2mNoV8yQTrFbRCRFDLjBC5UkA8jGe+a0ZW05L2P7Z/ZJtBfwm0+ziMnyN3zeZt5xt258znP41zh1C5ySqwL2XEY+Re6j2P+Pqcn9oXA+6sK44TEY+Re6jPY98+/qcnM+xFnsbcFrYaeumJePps8omuWmCypIu3y12BmBxjIOOe/rmqNjqsc8txPeW2nCWGzcRf6NGiu+4EHYAFLDJ7cgc5ql/aFwPurCuOExGPkXuoz2PfPv6nJ/aFwPurCuOExGPkXuoz2PfPv6nKuyte39XubOhXVvdz3V5eJp6SARJ5BW3hVhzukHmIyjGBlUAJzWtp9pp8U1zLaQWcsTalMsKSJbutxGNu1FaZhgc9U3HnntXIf2hcD7qwrjhMRj5F7qM9j3z7+pzctvEuqWUKx2rW6LG5eHNujmHIwQhYErnAz34qlLuTaRpRyWEXh3BsLWacxyi6WS4t4Xjl3N0Rk8zgbSAjAHGMdc5t/qMEdtbWcFvYtBJaRea6QIZA/BZt/3g3GMZx6iqjaldMxbEIbPynZkqO6jPY985zz6nKf2hcD7qwrjhMRj5F7qM9j3z7+pzPM+xWpteJ5NP+xFNPsrUQ+aPs9xFdQMwTB4KIiycjGfMJII65PPP6bJaRX6NqMfmW4zuXYXzxxwHQ9f9oU+a6lmiMbJCF6LtT7g67QfT9evqc1fKPtRfW41tYvaxNps08Z0mHyowvzjymjyc+jSyfzH0q1pm268OX2nwTwwXTzxykTTLEssahht3MQuQWBwTz+FY/lH2o8o+1MDptJit7e1gTOllkuWGom6eJyY+MeWSTkY3cx/Nn8Kr2lor+G9VEM9oqzTRtCkl3EjsqF8/KzA9x257ZrB8o+1HlH2obuFjqoLXTRHLdyS6d5UlpaLHGZEL+YGiEh2dVPD5JxnJ6jNaMtxocs1uhj0pYxcoWKJGPlFyy4J9PLAz6jk561wnlH2o8o+1VzC5dLHRpdWN7Lok1zFZRMNQZJ0hjWMCEGPbuA6jluTyeeTTdNaT+1tcj1BLWW6kiZWivJ/KSSTzkJBbcvPBPUdK57yj7VLcy3F3KJLmTzHChdxHJAGBn147mlcf9fidVPNb3F1dSWMmmvqA8hT9seJolQR4ZYzJ8pAYAZ5OBwTySsR0r7QW0z+y/s32x/tn2vZnyeMeWJPmxjf9z5s49q4/yj7UeUfajm1uK2ljetLRX8N6qIZ7RVmmjaFJLuJHZUL5+VmB7jtz2zVvUJNMHh1Us7S1kjaCPE5uoBJHJxuJj2ecTncMFiuDkcYxy3lH2o8o+1K+lh9bnc3SaMpsVkbT5PK1GBfNElv+9h53syxou1TgZDliPbvzmq3UF7o9nMsVpFcieZGS2jWMiMBNmQvXq2CeT6msnyj7VJbtLa3Ec8LKskbBlJAOCOhweKd7sErf16mh4n/5Dr7/APX+VF9o/wCuvlrv/HdnPvmsirSW095JJI0gZydzM7Elie+aZNaPA4VypJGeDUhdbEFFP8o+1HlH2oGMop/lH2p0Vs8sgRSoJ9aAIqKvPpF4trJcrGZIYiBI6KSseeBk4wMniqnlH2oEmnsMop/lH2o8o+1AxlFP8o+1HlH2oAZRT/KPtR5R9qAGUU/yj7UeUfagBlFP8o+1HlH2oAZRT/KPtR5R9qAGUU/yj7UeUfagBlFP8o+1HlH2oAZRT/KPtR5R9qAGUU/yj7UeUfagBlFP8o+1HlH2oAZRT/KPtR5R9qAGUU/yj7UeUfagBlFP8o+1HlH2oAZRT/KPtR5R9qAGUU/yj7UeUfagBlFP8o+1HlH2oAZRT/KPtR5R9qAGUU/yj7UeUfagBlFP8o+1HlH2oAZRT/KPtR5R9qAGUU/yj7UUAS0UUUAFXNO05tRafE8NulvF5skkxIAXcF7AknLDjFU619BuLW2TUDe+SyS24iSObfh2MisM7CGxhScg/wCBat1Ewbw7NEztc3lpBbLs23Llykm8bl2gKW6c8gYxzimjw9efbILZmiWSa6a1XLEgONvJIHT5xyKvHVd7SreNYT27FB9ndX2RMo2ps2sCQBwTk985OMvttcktrjz5pNPuJkuDOHljY+VKwA+UKQCuFHODjHHbJeNyeZ2MyPRS1jHcy31nB5quYo5XYM+3II4UgdOMkA5qObSjbWccs91bpNIiyJa/OZGVuh4XaMjnBYHFSz7Z7a3glvLdVtgy5UEnczZ/Ec9R0wevGbE91HcWkcdy1kZ4UWIXSiTzF2/dA2ttYAcbtvQd+Mq66j5lckuPBmqWxgEuxTNOludySIEduByygMOvKFh+YqO38OKdQsI59RtWt7qcwmWEucMpUMn3OvzDBwV96lk1GN7+G+I0+O8hmWWSZBITJIDnkbtpBPJKgdDjtmstwYoLWIXtun2SdplZVJIkbb/30vyjke/XjNKUb6k82n9f12EGhCfWmsLG+huD8xDRxTPjB6ECPcT9Bj3pzeGbmG8uYLu4t7UW7Rq0kokwS4ygAClgSM9QMY5xVldRhju7iYRaaiXEZiuYVEpVyWDZHz7uoHKkDg8HjOpYa1Cq3lxNqFpa3chhj2qZ0jCohUAGE7j0Xvt68dKUbWVxuWrsZNv4M1K6muI4GjfyZ2t96RysruvUAhDtHI5baPfg1gMpRyrDBU4IrpLq70+Oa4tovsl1YLOzwrceaWiZsA7SrAlTgdScY7n72YNLtSo363p8bd0KznafTIjIP4EilvsNNFyHQkh0m+lu5YXuktY5Y7dS/mRbpEwTxt5VumSeRxTb/wAI6np8Mck6r88qwkFHTa7dBllCt0PKkj36VFL4iuHtZYPs9qHliSGS5VW8x1QqVOScD7o6AZ784qC91Vb+UTTafaCcuHllTzAZj33DfgZ77QPbFU+UFewzVNNOlXZtpLiOWVeHVEkUofQh1X9M1NoGmQarqX2e6ult02OwzuyxCk8YU+mTnt05p81//bK29pNJaabbWqsYgfOdRkjIz87fh069zy60ji0q9iu7fVbG4KEgqqS4ORjBDKpwQTkjoPfGVtuNvQXStNtptVkga4tLxVhdkXdNGJDsJ+U7M5GM8gA4xUel6DLqkcbLdW1sJpvIiE2/944AJA2qcYyOuOtWrO9ttP1B7uKHTQ23ZsJmZYyQQdvz5YEE5JJxzjtnW0nUdNttN8iS9tbWN7hzdQK9yBsKqAsflt844Iy5IH0PIrP+vMnmMWDwnqFzpjX0O1otrumI5MOq5yd23aOh4Yg+3IzUutIks9Nt7u4uIlNyu+KHbJuZc9Qduz/x7Iq9Lc2r2gtZfscyQho4ZpfM82IEkgDYwDAEk5IOOeoxkW7jTSZNOjlsraOXatxKplYswOQcbip56sq8YOOoyrp7D5tTGtbaW8u4ra3XfLM4RFzjJJwK2Lrwne2dxbRTzQxi5ZkjeVJYgXAHy/OgPORg4289etQ29tBYXMd3b67ZGWFg6eXFMTuHT70YGPXJ6evSpjc2qXsd1FDptu8RO4IJXXeehwXOQDzkccHg8Zei3BsgXw7cCV47m4trWSKDz50mLAwruCgMAp+Y7gcDJx78UreHZomdrm8tILZdm25cuUk3jcu0BS3TnkDGOcVowanbsLmTVJLO4ma0Fu5YSHzn8xCNxVgWwF+8MdOc8ZiOq72lW8awnt2KD7O6vsiZRtTZtYEgDgnJ75ycZLxFzFNfDsqmT7Ze2dmI7g25MzsQXwDxtU8YPXp6046Lajw8t82p26TmZ4yhEhBwoO0YQjdz1zt5HNF3cNexOl1e253TmaRlQ5EhAGBjgrgdQOMHrxkt5ooNOksp5LOeIvu/eb8xSEYyu1hkYxknI46dMq6s/wCu3/BDmFl8KalDpH9oOn7vy1lKeXICFbGDuK7D1HAYnnpwcOv/AAjqenwxyTqvzyrCQUdNrt0GWUK3Q8qSPfpRc3UV1boty1l9pjRYzdr5nmArwvRtrDAALbe3c4JL25t76UTXKaes+8NPLH5uZX9CA2MHuVA747ZbcbhzFDVNNOlXZtpLiOWVeHVEkUofQh1X9M1DaWct7IUhaFSBkmadIh+bkD8K0NYvjd2ltCn2ZIrUsqwwM7BC3JwXZiRx2OM545yc+0vruwkMljdTWzsNpaGQoSPTIpKxV7rQbc2z2k5ilaJmAzmKVZF/76UkfrWhDZWVto8N/qSXE32mV0iiglWPAXGWLFW7tjGOx5rPubu4vZzNeXEtxKRgvK5Zj+Jq3a6sYLH7HcWdvewCTzEWfePLYjBwUZTzgZHTgU0DJItCmmt0kFxbxyyxtNDbOzeZIgzyMLt7HgkE46dKWeG3k8MW1zHaxxTi5aFpEZz5gCKckEkZyewFJHr00dsiC2tjNFG0UVyysZI0OflHzbT1OCQSM9elNbWA2jrp39nWgRW3iUGXfvwAW+/jOAOMY9qemv8AXX/INbksPhq8nvJ7ZJIN8N0LViWIBchjkHHT5D+lTv4SuleELe2TiZSwcOwVV8syAklRjKj/ABxQfFt0JPMjs7ON2uBcyMqPmSQBhk5b/aPAwPTHNVz4juzEsflw4WPyx8p6eSYvX+6c/X8qNLeYtbmpoPhsz65bWkl3E9tcx/aDcw7sCJd244YA5GxuCKGsNN1S+1C5jtnjt4dnkQG+jgwpLD5nkBBPHT1JxUfhrxA9lqdnPPEskNpEbdkXgyRMX3D64c/pVPW5obS7vLLTphc2cxQrMyFWIUkjjseea5lzLEO9+W2nbd3+e36dRfn/AMD/AIf8BLK3tJYtXhe0jJgheWGUzF2QhlAG5SEYcnnHNRQ6DPNbJJ9ot45pYmmitnZvMkQZ5HG0dDgEgnH0pmm6sNNhnj+wWtz56GN2mMmdpwdvyuB1A9/epE1+aO3RBa2xmijaKG4KsXiQ5+UfNtPU4JBIz16V0O34f5/8ArUd/wAI3dHRBqgkT7PhWYmKVdoLbc7igVsE/wAJJqe90i10y6tGt9QhuTJCkhiAk3HcuSRlANv45qvN4hnmsJLY2lqrSwR28k6q29kQrt/iwPujoBmkTUje/ZYZ7aASQoIxcjfvKgHAI3beP93PFOVuhLvy6/1/Wp3Gq6fa23gPW7zTVZLLUILSeKN23GI+cVZM98EHB9MV5lXo+r6xYS+C9X0yzuF+z2sNrBa+Ydj3BExZ3CnnkknHYYzXnFcuG5/ZLn3897dL+dt/MvTp/T6/iFFFFdABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAEG4+p/OjcfU/nSUUALuPqfzo3H1P50lbfhqLzG1JkhtJZo7TdF9s2bEbzYxuy/wAucE9fpznFNK4jF3H1P50bj6n866+Q2bS3B0v+yf7RHkiXzxCID8n7zyw/7v7+On/AeM061ttHutVt5IJdOjtINTdpvPkVA0JEe0ANyy5DgDnGecU+XUV9Djtx9T+dG4+p/Ott9Rhs9Hs47e3sJnkWUT+ZAjvyxC8/eXA5BBB/Spb4INIt/sB0/wCwmCPz/wDU/afMz8/3v3n3v7vGMds0rXKejsc/uPqfzp0aySyLHEHd3IVVUEliegArtL2TSP7RsUisrEWX26LZcfardsRZ5DIqK+CMZ8wkjHXJOaMGr20cmj35jsIZo72RZlito/lhBjK5XHbLYY/N71Sim7E3dv68/wDI5ht6MVbcrA4IPUGk3H1P511sQ3+JLiW/l0ttsTtAkLWapN8/ALFWQNgk5YbuMd6vxWumHUtSuLUWawg2wWXZbyxgmMmRR5pSPJI7YIxwoGcTFXSY27XOD3H1P50bj6n867eF7GKS9M+l2f2v7Y/nWktxbQBIsDYql0II+9zGVPf0NcQ5Bdio2jPAznFIZLcQT2sipOCjMiyAbs/KyhlPHsRUW4+p/Ouovb60ubC7gkFkfK061MMgVfMMoEQYb/vEgbgVzjjpnJq/Fa6PBqM8txJpbwT6mrwoJY22wbZDyB91clRjgggcDiqcf6+ZN9P67HEruZgqkkk4HNS3lrc6feS2l2pjnhba6bgcH6jiun1Ke1ubeyOnJpSXYaJj8kCKSYQX3bvlxu7HgH3pZ7W0h8Rao9s2lF5Iy9gGmgaDO5QeM7Adu7AbH06Ucutgvpc55NJ1CS+FmkRM5i87ZvX7mzfnOcfd5qqsczQvKqSGNCA7gHCk9MntnBrsd8N34unLT6VNatZxpLJK0CoWEAA2b8Y+cfw4x3xVHR7qe30nUtMjfTBch43T7QLZkcAnd+8fKt1GOT3x3osO+hzO4+p/OjcfU/nXV28mnR+GV2WdpcyGKT7SXureJ1ky2CFZPMOBtI2MAcYx1zyi43Dd0zzSa1sHQNx9T+dKgkkkVIwzuxAVV5JPoK2NUudDlsdumW3lz7h832d0478m4cf+O/jTPDVxFb6ufOlWBpIJYop2bAikZCqsT2GT17ZzRbUL6FG8sr7T5Fjv7a4tXYblWeNkJHrg06x0+71JpEs9jPGpcq0yISACTgMRu4B4Ga1NMsRZS3STvpzXxhDWhluIZYc7huydxTdtzgN/PFXrKO3bxYJoZrCFY7Yi4ZbiOKPzWiZTsyQCN393j04p2C5ye4+p/OjcfU/nXUaJp9mXsFvZtNT7LqDG7M08ZDR4jwM5O9chhxkDPOOtaFlcaMunwRSR6WSYoQ7OkZfLCXecnnPCfTjoTmi2lwucPuPqfzqcWl0beKYRuUmcxxYPLsMZAHU9RW/dXVhJpt3arFYKI9PtnhdI0EjTfu9/z9SeWyM9jxnJqG6cR65oTsyparBbGNs4UDjef++9+fxp8qvbz/r8hX0v/XT/ADM2XR9Tt54ory0ms2mJEZux5Ctjr8z4H61XvLa4sLyW1uhsmiba6hg2D9Rwau38keqasY7O0s7UmR/nWdlWTnOS0jlR+GOtbt7HB/wkmqzJNps1xMu+yeWeGSI/MA2ckoG25wH/AJ4qVqkxvR2OP3H1P509Y5mheVUkMaEB3AOFJ6ZPbODXYWY00aldTFtPZ1jhWWENbpGXIPmMhkRlwCOiDnPHAFMivUj/ALe07Sn0xU+1B7UXCQbHQO2cPIMNwRjJ9cU7Bc5U21wLJbsq3kNIYw4OQGAzg+nB79efSodx9T+da+m5/sXW/Px5Plx4xjHneYNuO33fM/DNY9SMXcfU/nRuPqfzpKKAHrLImdkjLnrhsUjSO5y7sx9Sc02igBdx9T+dG4+p/OkooAXcfU/nSrI6tlWYH1BptFAD2mlcYeR2HoWJpu4+p/OkooAXcfU/nRuPqfzpKKAF3H1P50bj6n86SigBdx9T+dG4+p/OkooAXcfU/nRuPqfzpKKAF3H1P50bj6n86SigBdx9T+dG4+p/OkooAXcfU/nRuPqfzpKKAF3H1P50bj6n86SigBdx9T+dG4+p/OkooAXcfU/nRuPqfzpKKAF3H1P50bj6n86SigBdx9T+dG4+p/OkooAXcfU/nRuPqfzpKKAF3H1P50bj6n86SigBdx9T+dG4+p/OkooAXcfU/nRuPqfzpKKAF3H1P50bj6n86SigBdx9T+dG4+p/OkooAXcfU/nRuPqfzpKKAF3H1P50bj6n86SigBdx9T+dG4+p/OkooAXcfU/nRSUUAFFFFABVqz1G4sEmW2Kr56hHJUHK5zt57EgZ+n1qrWno1hDfx6gJ3jiMNr5iSysQsZ8xASccngnjB68DOKdridupD/ak4+6kK44TEY+Re6jPY98+/qcn9qTj7qQrjhMRj5F7qM9j3z7+pzpWGhLHrL2t49rcf6MZYlMsiLODGWVlZUJGOuGx0xVOx0OW+gila6trbz5DFbrOzAzMMZAwpA5IGWwOevWlyk2iQ/2pOPupCuOExGPkXuoz2PfPv6nJ/ak4+6kK44TEY+Re6jPY98+/qc6cPg69ltVmN1Zxsyqwid23fNu2jhSMkow69vTmqzeG7waAdXRle3VQzDypVwC237zIEPJx8rGjlHaJV/tScfdSFccJiMfIvdRnse+ff1OT+1Jx91IVxwmIx8i91Gex759/U5nv9Bm0+e3t5biFrm424hCyAqG6EllCkdsgkVI/hq5GtppUE8U90d29UjlBjwMnIZATxz8oOe2eKOXoK0Sp/ak4+6kK44TEY+Re6jPY98+/qc2bbxLqFpBJDb/Z1idgfLe3SRVwCMAMD2Y+/J9TmLWNEutEuI4rvB81N6MEdMjJH3XVWHIPUCjRtFn1ueSK2cK6KGIMMsmRnH/LNGx+OKEtdBuMVqFxrl5dXElxMY2ldid5QEgHqoz2P+PqcuXXrpFCpBp6qOADp0DED6shJ+pJNXbPwXqt606oiq0M7W/3JHDSL1GUUhRyOWKjnrwcU00JzZJcT39nbmUOY4pXYM5QkEcKQORxkgHNCVtgsiA6rPklUhXsuIx8i91Hsf8AH1OT+1Jx91IVxwmIx8i91Gex759/U5sXFjCnhq0uYhDLLJcMjyRu+4fKp2FSoHGc5BOc1cufBOq2ptxNsUzTpbnckiBHY4GSyAMOvKFh+YycnkK0dzL/ALUnH3UhXHCYjHyL3UZ7Hvn39Tk/tScfdSFccJiMfIvdRnse+ff1Obo8MzvNbpDfWUqT+aPNWRgiGJdzgkqO3cZB9adD4UvLm8jhtJobiOS3+0rPCkjrs3bfuhN+dwxjb79OaOUdolD+1Jx91IVxwmIx8i91Gex759/U5P7UnH3UhXHCYjHyL3UZ7Hvn39TlNU0y40jUHtLsASKA3AIyCMg4IBHB6EA+oqWz0aS70ye/a4it7eFwjPIsjZYjgfIrY/HApWQcsSP+1Jx91IVxwmIx8i91Gex759/U5pVoQ6NcTiDY8Q862kulyTwqb8g8dfkOPwp82iG3sVnlv7NZGgWdbcu3mMhx0+XbnnpnPFO1hq3QzKKltoGuruK3jIDyuqKW6Ak45rRfQSt5DbR6hZzyySmFlh8xzGw7FQmT6AqGFOzC5k0VtXPha9s9Q+zXEkUai3+0tPIroqR525Ksocc8Y25PGM0aRplrPq0sDXFpeqkLsi75o1lOwn5TszlcZwwAOMUWAxaK1tM8PTanbxSx3drAZ5zbwpMzBpHAU4ACn+8OTgeuOKdZ+Gry/wBJmv7dlKQq7Ohil4CjJ+fZszjnG79aLMLmPUrXUz2iWzvuhjYsikD5ScZwevOBxWhc2UMXhm1uo/JkkkuGV5UkfcvyqdhUqF4znIJzmnz+G54rhbWO9s57xpUi+zRO28Fuh5UAjpnBOM84os72C6tcx6K1tZ8O3uiRwyXWGjmJVW8uSPkYyMSKp7jkDHvVGzspb6UxwtApAyTNcJEPzcgfhSGV6KmurWSznMUrRMwAOYZklX/vpSR+taEFlY2ujQ3+ppcT/aZXSKKCVY8BAMsWKt3YDGOx5oAzzdztZraGQ+QrmQIBgFiMZPqcevSoa1YtAmmt0kFxbxyzRtNDayM3mSIM8jC7ex4JBOOnSlnhtpPC1tdR2scNwLpoXkRnPmAIpyQWIzknoBTs/wCvuEZNFbMPhi9nvJ7ZJIPMguxaMSxALkMcg46fIefpUz+ELpXhC31i6zKWDiRgqr5ZkBJKjGVB/riiztcLowKK2I9Axqtpby3cMltcRG4+0QbseUu4sQGAORsbgiptK06x1P8AtC5e3ZI4WTyoRfR24AYnrJICD0/Gi2tguYNFbdjb2csWsQvZxk28DywymYu8ZDqANykIwwTzt57VDDoE81rHJ9pto5pommhtXZvMkQZyRgbR0OASCcdOmS39f16DMqitj/hGbs6GNVEiG2wrOTFKu0FtudxQK2Ceikml1vR7TSxbtbajDcmSGOQxASbjuXJIygG305zQ1bcSd9jGorR1i0hglguLNSlreQiaNCc+XyVZc98MpA9sVnUhhRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFaWjXSWrXJmW2eGaIROlxvw2WDDGxg3VRk54HrwKzafFDJPII4I3kcgkKiknAGTwPagT1Oig1ZYNUa+l/s6WQRiFUYS7YBt2hUCsMrtOCxz0J5OCWwalHbKq400rBKZIQyyP9mkOPufN8y5APzbhweDxu5yijXuLl8zqI9duIlQPd2RMYRWJRidy78dDyP3jZI9Ppujm1XzdPktpH09Gkgjtpp0Ry5CFSgHzYx8oyVA6Hrxnm6KNe4crve50M99FLp8ViPsMFrFMJZIo2mcNJjH8TnggnJXH44XMrawwe02S2MEFmrx/Z18yRD5gwy/MzEqR1wQByRzgnCk0+5hjtZJUCJdjdCzOoDDdtyeeOR3xRfWFxptwIbtUVygcbJFdSpGQQykg/nRqLl8y9Pa2N5NuW703Tdq7WjUXDKT6g4kz9cgenqbNlPDpUM0AvNMv4JmRzuSb5HXO04IQnqc9Rjseh5+prm1ms5hFcpscorgZB+VlDA8exFPVbDtfQ3pdUS4M/9o/2ddLJO8xVxL+5kfrsKMCVOBnkgbfpuz5lSa0treW9t1S1DJlQSdzNn/gQ56jpg9eM5dFTqFn3OgN7ANHTT9unxrG+8zIZjIspAG4fPg8AZIGBg4HTL5dRjfUIb9hp0d5DOs0s8YlJkkBzyu7aVJ5JUDocds85Vq902509YXuVj2TrvjaOZJAwzg8qT37U9dxcvS5pwXhtraOAXdrthM4J2sSGmQIeh5AA6joQevGZIdRMMcUU09lLBFbG1lhdXKuC5cZ2kE4Y53KRjHf+LnqmmtZreOCSZNq3CeZGcg7lyVz7cqaNe47PuaEllZXMzyDUdPsOcGALOyqf9khXyPcnrntilstTHh+7ZrFbO8mXlLxTOuMjoASgI9mUg+4rPs7KfULkQWqhnKljucIoAGSSxIAAA6k0yeFred4pChZDgmORXX8GUkH8DT2Gadt4intbFbdbS0d1gkt1ndGLrG+7cB82OrE5xn3xxVKa5bUJ7cTtHCEjSEPg4VRxk4yfrgfhVWnvDJHGkjxuqSAlGKkBgDg4Pfmi99xmpBZ2tlcRXMeuWErwuHVFinOSDkdYwMZ689KfZzCx1Q332q1dzvWVGVmUlwQQMHJXDH5geMH2zi0UtSWjpW1b99GS2nJBHbGze2VHMbKWLBfvbiMnO8HPy9+N0dje2unajJdxQ6aDt2eWTMyxEqVO3D5YEE5JJxzjtnCgt5rqdYbWKSaVzhY41LM30A60tzaXFlOYLyCW3lXrHKhVh+Bo13Dl8zahvhaz2rwzWUK2d01wsah2VZG2/LyxLJ8g5z69eMzQat9nskty+ns0UElt55Ry6LJuyBhtpHzH5sZHPXjOJb6bc3dnPcwLG0duu6QGZAwHHO0ncRyOQKq0a7XC3mb73dsdFXTWTTlRG3NKpmLiUgDcPnweBycYGDgdMxPcMdZGp/b7eK5SQOSikhXHTA53L6n2PXjOLRRre9w5dLXNLVnt5gslvBZQMrESratJyTz/ABs2R7rgfoaq2d/eafKZLC6ntZGG0tDIUJHpkUGwug1uogdnuhmFFGWcZwMKOeSOPWnTaZf290LaeyuIrgruETxMrkcnOCM44P5UK6H6kd1d3N9OZr24luJSADJM5diPqat2mrmCw+xXNnbXtuJPMjW43jy2IwdpRlPOBkHjgVnVPbWVxdx3ElvHvW2j82U7gNq5Azz15I6Uxl6PxBPHbRoLW2aaKNoobllYyRIc/KPm2n7xwSCRnr0prayG0VdN/s60CK28TAy79+AC338ZIA4xj2rMoouwsb58X3Yk8yOysona4FzIyo+ZZArDJy3+0eBgemOagPiW8MSx+XBhYvKHynp5Jh9eu05+v5Vj1JBbzXU6w2sUk0rnCxxqWZvoB1ou3oKyWpo2ettFeWD3MSvDawtblF4LxsW3de+Hb9KrSXAtY7yytZEnt5nU+aVIJCk4OD0681Fd2N3p8oiv7Wa1kZdwSaMoSPXB7cVBRcZpaZrA0yGeMafa3P2hDG7TmTOw4O35XUdQD6+9SR+IJo7dEFramaKNoobkqxkiRs/KPm2n7xAJBIz16Vk1PHZXEtjNeJHm3hdUkfcPlLZwMdexou2LQ0JvEc82nyWptLVWlgjt5J1Vt7IhUr/FgfdHQDNV7rUzfWlvBPbW4khRYxdDfvKDoCN23gf7OeKoU6ON5pVihRpJHIVUUZLE9AB3ou2Gxf1i7huJbe3syWtrOEQxuRgvyWZsdssxI9sVnVIbeZbcTtFIIWYoJCp2lh1GfXnpUdIYUVJb28t3cx29shklkbaqjuaVLaWTzdgDeSpZ8MOBnGffr2oAioqZLK6kgM0dtM8QBJkWMlRjrzUNABRRSujRuUkUqw6qwwRQAlFFFABRU8NheXEe+3tZ5UzjckZYfmKhYFWKsCCDgg9qLgJRRRQAUUAZIA6mrAsLg3ptCqrOp2lWkVefTJOKLgV6KnisrqdnEFtLKUOG8tC2D+FQUroAoqVLW4kgaaOCV4l+9IqEqPqaipgFFTT2stssZlC4kGVKurAj8CahAJOByaNwCiprm1ltH2TbA3cLIrEexwTg/WoaNwCilCMUZwpKrjcwHAz0pKACiiigAopVVnYKgLMxwABkk0hBBIIwR1BoAKKKKACiipDBILZbgr+6ZigbI5I5/rQBHRU09rLbLGZQuJBlSrqwI/AmoaACinyQvEsbOuBIu5DnqMkfzBplABRRRQAUVM9rLHbJOwXy3OAVdTz6EA5H4017eSOOJ3UBZgShyOQDj8OaLgR0VZbTrpNS/s+SLZc7xHsZgOT056c569KgkjeKRo5VZHQlWVhggjqDQA2iiigAooooAK2vCl+LDXYy7QIkqPGXnjRgpKkDlgdoyRk8cdeKxaKadmJ6nQLdNZw6lLdR6W98DCItkUEiAc5KqoKHjGeD781p+Gn0253XGorpypcXZE0LGCJYUwOQHRnIJJGEIxjqO3GUUJg1c6G3k046St7ItsLizikgEBC5mZj+7crj5sBmyf8AYXPWr1xHpi+EJUL2csy20TwyrJAJDIWXcu1UEmQCQd7HOM49OQoovoFtbnRa1ZmXTdMkDaYLjYY5xbT26/MXO3cEP93Hzdu5qHxNBsksnE1rKq2kUR8i5jlIZUAIIVjj69Kw6ntLyWymMkKwsxXbiaBJR+TgjPvQ3f8AMFoaPh5YTJeE/ZPtYg/0UXhTy924Zzv+XO3djdx+OK2NTjtNUvLtUu9OMwNkRMZEjTYsJEm0nHAO3KgZ46cVzN5qM18EE6Wy7M48i1ih/PYoz+NVaalYLF/XPI/4SC/+x+X9n+0yeV5WNm3ccYxxjFalisEnhWVH+xWrKsj+c7W8skx7JsP71DxgFePbqay7bWLm0t1hiismVc4MtjDI34syEn86pzStPM8rhAznJCIEX8AAAPoKnZWDd3OnvPsI0ucR/wBnfYTap9l2eX9p875c7sfvOu/O75cdO1U9ZtWi8P6UPPtJGgjdZEiu4pGUtISOFYnoawaltrh7W4WaIRsy9BLEsi/irAg/iKpu4JWRe8Pi2bV1+1+TgRuYhcECMybTs3Z4xux149eK6Xz7GS8tBfPp9zdRaaVRIpII4RL5zErkqYgdpPVcHsckGuYk128kXBSzT3isYYz9MqgOPbvUf9qTj7qQrjhMRj5F7qM9j3z7+pyuZ9P6/q5LTudBbatHaaxqnlW9jaI1k+EPkXAaTZ2cLtOT/CuF7Yp/hY6fLGZtUaxYT3RE8cpt4vKTA+YBkLEHJGIyuMdR1HOf2pOPupCuOExGPkXuoz2PfPv6nJ/ak4+6kK44TEY+Re6jPY98+/qcnMwszbsbWwks4Ll5bBUisLlJEkkQO02ZNnynknBTDY7DnOBTL+/mvvCdkIm07bAjxzp5dukqnfkbRgPyCOV68571j/2pOPupCuOExGPkXuoz2PfPv6nLX1KZ42QLGgIKrtTBRT1UH0P59fU5TbY9b3sQ2rQpeRNdLuhDguu0tkd+Ayk/mPqK0dYuNImjiGk2/lMCd58h48j/AIFNJn8hWTRT6WK63NnQJIzb6naCaO3ubq22QySyCNThgzIWPA3AEc8du9WtMtfsltcxRtpbamsqYN1NBJH5RBztZiUJzjPfHTvXOUU7isdVaW9tLfa62nz2cVtNC8MAkuki3MWU4UOwOODgmmWNpYtZQXE0unqsOn3McqPKm9psybPl6k4K4bpwOc4FcxRSvp8v8/8AMDuL640ZtNuIYY9LB2OEZEj38RREYPXO8vz9R0GKzdeurC5ttSWCKwj8jUFW1+zRohMWHycrywyFOTkDPGOlczRTcru4JWR015PdW/jm5e0+zHO5YluJVjiaAphRuLLgFCMYIPpVixlsNH8SWhtJLaL7RA4njkkiuY4HIYKBLggD7vfgHBJ5rlprqa4jhSZ96wpsjyBkLnOM9ccmoqVwsdAt01nDqUt1Hpb3wMIi2RQSIBzkqqgoeMZ4PvzUmk/8TK21ZpU0qBZ42MSSNBEVlyuNm8hlGM9Dtrm6KL6WCx09pcWKWNtZTR6fhtPuTNIyIX80GTYN/UHhcYIzkdRir1/bR23hl0trS2kkNlCzJ5Nv5kOQpeTJbzjnPdQBuyDgCuKrRm1/UZ7Q28kse1o1iZ1gjWR0XGFZwu4jgcE9hTbun/Xf/MLWf9eRWnspLe0trhmRkuQxUKeRg4Oat+HZ4rfXYJZzGI1WTPmNhT8jcE5HXpWfLPLOwaeV5CqhQXYnAHQc9qZURutypWexvaXPaXM928Npp9pdiEC1jmbMJbcNxPnMy7tucbjj8cVaubuytFv5rePTnuwluOIkkjEmD5hRSCpH0BX07Vy9FVcmx1+j3WnS2Etw+n2Mt1Jcs08DzwW6hMDaF81WwM7vuFSPyxRtbcXPhS8B/s1GEqPbh5oEmCgtvGSQ57cHr2rnqKG7hY7a+to7bw0yW9pbSSGzhZkMNv5kOQpeTJbzjnPdQBuyDgCj7dpr6yWVdOt47XWofs7wIkf7nLbjuHLLwpycgZ4wK5qbX9RntDbySx7WjWJnWCNZHRcYVnC7iOBwT2FZ1U5a6f1qLl92z/rQ6yOTTbrWLC5kliOnLE0UNpLKgMEgU4DZBGC+G3lSpzyOoolFtqGtDTl0+1gkvLbykkjngmAl3ZRt0ShV6bSABwcmuTqzaX9xY+b9ldY2lQoz+WpYAjB2sRlcgkcYpXXUb3ubcUsMmoawumpGohsGhtiigF1UqHfjqzIHJ+prnFdkDBGKhhhsHGR6Gn29xLaXMc9s5jljbcrDsaYzFmLHGSc8DA/Kpeoy9JcRDS7aIwwyvtfLFm3J83oCB+Yq3ItqNFdS0DuIkZHDRhi2RkYC7uASOTz/ACoJqU8caqojyoCqxjGVXuPoe/4+py7+1Jx91IVxwmIx8i91Gex759/U5jlZFmTX7+U4a1+xm1VgYtqxs2MfxA/N9d3f8KtTyxT60stybZ7d9zRlDGMnbxuwMjnH3hj8M1n/ANqTj7qQrjhMRj5F7qM9j3z7+pyf2pOPupCuOExGPkXuoz2PfPv6nK5WKzLcwhnmmjEVvDI1vhD58RDNvBzlQFU4yO360k8lvbNcmD7LI6pCEO1XGdvzEA8HnrxVX+1Jx91IVxwmIx8i91Gex759/U5P7UnH3UhXHCYjHyL3UZ7Hvn39Tk5WOzEhnV1v3YRxmSHhV4GfMQ4A/A8U/wC1JBYW6xxW7syuJd0YZupxz1H4YqnNL50pfaqZ6KgwAKZVcpVjbcWqaMwzbvIsSNG+Y9xbIyNoXdwCR8x59PSKUWZkjMbQoLqZXJwG8he4x25J49FHY1k0UuUXLobWoeUTam1Fqs+9wwLwsuMDGcAJ69aZJAq+I0eJ7ZYfNWQFZkChQRnvgfSsiihRsHLpY2DAXt7mFZ7dZPtSyAm4QDbhuQc89R05qcfZL3UEn8y3Ea3bNK0pC70wuDg8kEhvz5rAoo5AauasSpcWKre+SscSN5UqTrvXqQpTJJGT6A++KdP9nFpJt+y/Z/JXysbfN8zjOcfN/e68fpWRRRyjtqaN7CU02z/eQsY1YMEmRiMsSOAc9KiiBsNQhIkgk+6Sww6gH6jGR+lU6lt7iS2kLxEBiMHKg8fj+X0zTtYVtLGmkyXOtXUkjW5C7/LG2JBJ83HLKVzznJGeKfKLb7ZdGz+y+btjMfmFCnT58ZAXOfYd8CqP9qTj7qQrjhMRj5F7qM9j3z7+pyf2pOPupCuOExGPkXuoz2PfPv6nM8rFZlqzuJfs15bg2YmLqwDrEFPJzgn5T14/SnRNarpI2wQytsfzd00aMGycYBG48YxtOD+dU/7UnH3UhXHCYjHyL3UZ7Hvn39Tk/tScfdSFccJiMfIvdRnse+ff1OTlCzLEkNuLOeYPb4a2iEahl3b/AJN3HUHg/XNQ6jdI00sEMVv5QYFGjjAI4/vDk596b/ak4+6kK44TEY+Re6jPY98+/qcodTlIwIrfA4QGFWCD+6Ac8fXP6nLUWCuM0+4+zahBKdu1XG4sgbAzz1q7JJsvriS4+xuRExi2CMqTu44Xgn681QmvJZ49jrCBnPyQIh/MAGoKbjfcqxr6fLFNJNPOtsrAIvl4ijBHdvmUjt0UZOaU/YTcz58kR21w0iAEfvU5wue/IH4Max6KXLqLlNu0FqdLbzzbsXikY5MalH52qBt39geCBzj61pYt+iRMTa+Yjk/K8YfZgYyAcnnPXms2ijlCxo3sJTTbP95CxjVgwSZGIyxI4Bz0qDTREb0ed5f3W2eYRt3YO3OeMZx14qCKRoZA6BSR0DoGH5HipJryWePY6wgZz8kCIfzABp2drDs9jUuNlwqxPJamb7HtG1kVFfzc4BGFBxn6/jWfqaxJe7YGjZBGgzF90nYM/rmqlTwXcluhWNYSCc/vIEc/mwNJRtsCVi7pYja3kSTyItzczyGJiox02PyfqvP16U+D7OLSP/j18jym87ft8zfzjGfm/u4xx696y5ZWmkLuEBPZECj8gAKZQ43FY0WhP9gr+8h3CYyFfOTdtKgdM5/DrWezsyqrMSFGFBPTvxSUoO1gRjIOeRmqSH0NbXP+PbSfN/4+fsK+Z643N5effZt/DFHif/kOvv8A9f5UX2j/AK6+Wu/8d2c++aotqN0+pfb3l33O8P5jKDyOnB447DtUDyPLI0krM7sSzMxyST1JNU3cENooopDCiiigArU8P6Xb6tqf2e6u1tk8t2Gd2WIUnjCn0yc9s45rLqzp99Lp19HdQBWeMn5XGVYEYIPsQSKa3Ey1Doqz/aXTU7IW9ts33DeaFO7OMDZuPIx0/TmrOneFLvVJriKxnhmaCTZmNJXR/Qh1QqAf9or74qhJqW6G5hgtYLaG4KFo4y5C7c4wWYnnPcmrWl+Ip9KhhjjtLWc285uIXmViY3IAPRgDwo6g47YoVuoO/QiTQ7mT7OyyReXPBJP5mTtjVM7g3HBG3tnqPWpW8N3g0A6ujK9uqhmHlSrgFtv3mQIeTj5WNV49YuItJm09RH5UzElyDuUEgkA56Equfp9c2JvEc82nyWptLVWlgjt5J1Vt7IhUr/FgfdHQDNGlg6iapaQWVppcsKW8wliZnkjkkImIcjkMFK46cemc03X4LeC8tzaW6W6TWkUxjRmKhmXJxuJP60Xetpd2lrbtpVlGlqfkKNNnGclTmQ8En6+hFSXN9Frs0S3CafpfkxBFlxOQyqAFXq/bvge5pvXbuCKWn6c+oNLiWK3igTzJZpidqDIA6Ak5JAwAetaOt6KltcyNZNH9niW1QkFvnaSLduGRnBKsecdRxUUc0ei+YILqw1WK5Ty5oAkwXAIIzlUPUDGD2p58T3ElzNLPZWUwl8o+U8bBEaJdqMAGHbPByDnpQuXqLX+vkZl9aSafqFxZzFWkt5GjYocgkHBxVoaNINEXU5riKKJ2ZI1ZZCZCOwIUqD7Eg1LLFFq9xNqF3q1hazXMjSPC0c+VJJP8MbDH4mkg1Y6Xb3NpZw2sjSo0L3aGXMiH/ZLBcemU469eanpqProE2gTw2skjXNsZ4olmltAzeZGhxgnjb/EDgMSM9OtLq8NsNN0u5trWO2e4icyLGzkMVcqD8zHHApJ9fnntXRra2WeWJYZbpVbzJEGMA5bb2HIAJx165Zf6yt9p8Fp/Z1pbrbgiN4jLuAJJI+ZyOSfSm7dAV+pStrZ7u4WGJolZuhllWNf++mIA/On3ljLYyKkzwMWGR5FxHMPxKMQKZbXVxZXCz2c8tvMv3ZInKsPxFPvNQvdRkV9Qu57p1GFaeVnIHoMmgZNp+lm+guJ2ure1ht9vmPOW/iyBgKpJ6dhWjY+DtR1HzTaPFLGkphSSNZJElYY6MqEKORy20c+xxjxXkkNjcWqhSlwULEjkbc4x+dWbbVvJsBZ3NlbXsKuZIxPvBjJABwUZTzgcHI4o0FqTw29u/hm/aS0jW6tZolE4Z9xDFsgjdt7DoKX/AIRm7OhjVRIhtsKzkxSrtBbbncUCtgnopJqK11lbbSZrD+zbSVZyDJI5l3kjO08OBxk9vrmpJvEc82nyWptLVWlgjt5J1Vt7IhUr/FgfdHQDNPQNbj9Z0Wz05rb7PqlvL50MTsu2TI3DJb/Vgbfb73tTr/RRJq8NrYi3ijNokzSrK7R7dmWkJZQw9cY9hmqc+rfare2juLK2kkt1VBMTIGdF6K2GxjHGQAeOtWpfEskl1DPHp1lCY4fIZU8wrLFt27GDOeMdxg+9Dtr6/wCf9f0g1/AavhyV2LLf2Rthbm4+1bnCFQ4Q8Fd2QT025PbPGXL4ZuTdXcT3NukdpEk0k+JHQowBUgKhbGDnJAx3xUE+uTSwyQR29vBbvB5CwxhsIu8OSCSSSWXqSf5U6LX5E1Fb6Wzt5p0SNI2Lyp5exQoIKOpzgDP9KNA1/r5f8Es2WgR6lpdt9nubaK5lvJYEeV3HnYWMqqgA46tyQBzyelVobGE+Gbu6xDJPHNGpPmOHhB3cbdu1gcdc5GPekTxBdreQXJSFpIbx7wZUgF225BAPT5RwPekttaFvpc9idNtJVnIaSRzLvJGdp4cDjJ7fXNLS39eQf1+f/AJJvDs8Nm04u7WSRLdLlrdGYyLGwXBPy4/iGRnPfGOaiv7a3sIbKEpvuHjW4nfceA4BVAP93BJ65b2q3qeuRvD5FhBAvmWkEE1yA/mOFRNy8nAG5ccAZx1PenqV1BfW1pOrbbmOJYJo8HkIMK4PT7uAR6j3pytrb+v60/4YFe2v9bf8EVI7G9v4YbeGSJCzlsvgkZJUZOQOMDNMNgJ75be2HlkpnHm+fn8Y1P5YqGwZUut7SpFtU4LqSCSMdvr/APr6VbtUhtGk3XVtIrp5cqMGIzkEAbSCRkdQex68Zy1RLdmQPpcsU80c8kcKw7d8j7sfMMjjGcn0x9adBp8EtncStexKY3VQSHxg55Py55xx+tW2mRpJWnuLNkdUSSLa20Mowu3ackepBx168ZjhEEHnK89o0UhAkiO8qrZ+XaQckD1z69eMq7sHMQRaNdTWf2lANm1mX5W+YDOTnG0dD1IP6VC1jIqyksn7uJJTyejbcfj8wq6RAbZYZ5rR/LBQOQ5aMk5AGDhgCTzzjnrxl7SobV4XuLNS0awyyKrFjtxtA5wRwMkDsevGS7DmKc1itox8+eKUowDwxOQ+D7lcfzxUk9nBJdxW1ksiO6K5aaUEcoGxwox9aZexrI0lwbqF5MjeiZ+96L6jHf2PtmEXsgukn2ruRAgGDjAXb/KnqNakkenGa5SGC5il3qWDRq7dO2Au79KLjTJ7cyh9paJlVlGc/MMg4I/+v7VFa3RtWk/dpKsibHR84IyD2IPUDvUw1WZb1rkJEGZFTbtO0bQMcZ7bQaPeuGpJBotxczTRROjPE+wgK5BP1C4H4kVEluh0qab920iuo+8wZBz2xg5+valtNUktFQCGGUxyeajSAkqxxnoRnoOtNivhFaSW/wBlhcSHLOxfJIzjo2OM+lHvBqP8qGTTIJBCqSGcxsyk5YYB5ySM89qnl0yCaaaGy8xJIZxCfOcMGy20HIAxz2561VF/izW3FrBhW3h8vu3cc/ex29MVJJq8ry+ZFDDAxlEzeWD87A5Gck/kOKGpX0DUZa2KyGF7mVY4pGdR82CSoBxk8DOQM0+XT2Nw6iNrSONAzNcPuAGcAgqvIJ6YBpsmoiVYkaztxFGzMIxvAJbGcndnt60v9qPwnkQ+QE2eR823Gc9c5znnOaPeDUfaWiR6h9muY451kjLK4Zum0kEYI/UVDaS2SRkXcDSPvyCpI+XaeOo77T+FOj1No7ozm3hdguxFbcAi4xgYI7euaqSMHkLKixgnhVzgfmSaLN7jL9taQXmh3bohW7syJS244kiJCkY7FSVP0J9BWdWjbXcFnol0kblru8IiYBTiOIEMee5YhenYH1rOrRggooopDCiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigD/2Q==)

Figure 1.4 Declarative base partie 1

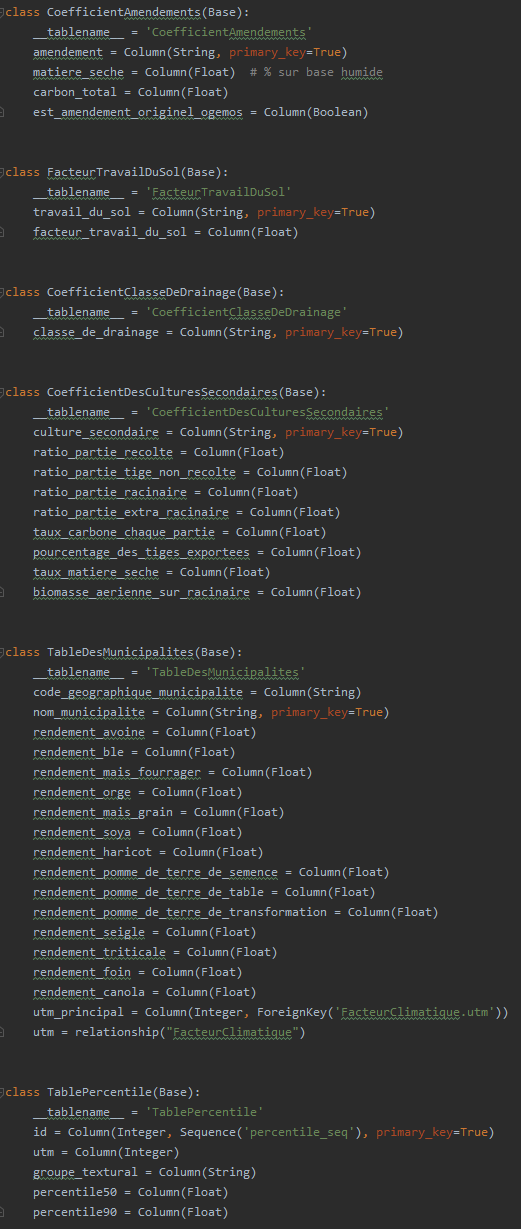


Figure 1.5 Declarative base partie 2

Pour la modification ou l’ajout de données, il y a différentes façons de faire et des options qui répondent à différents besoins. Pour ce qui est de l’ajout, il s’agit de prendre la classe créée dans la base déclarative et de créer un objet qui correspond avec les paramètres de chaque colonne en respectant le type. Il faut ensuite l’ajouter à la session puis la commit et la fermer. La modification de données s’effectue en allant chercher la donnée que l’on souhaite modifier, la modifier, commit et fermer la session. Pour plus de détail sur la façon d’effectuer ces manipulations, voir la documentation de SQLAlchemy au lien suivant : <https://docs.sqlalchemy.org/en/13/orm/tutorial.html>