Перепишите программу, реализованную в прошлой практической работе с использованием Jetpack Compose

package com.example.practice9  
  
import android.annotation.SuppressLint  
import android.app.DatePickerDialog  
import android.content.Context  
import android.os.Bundle  
import android.provider.Settings.Global.getString  
import android.widget.DatePicker  
import androidx.activity.ComponentActivity  
import androidx.activity.compose.setContent  
import androidx.compose.foundation.background  
import androidx.compose.foundation.layout.\*  
import androidx.compose.material.\*  
import androidx.compose.material3.Button  
import androidx.compose.material3.ExperimentalMaterial3Api  
import androidx.compose.material3.Scaffold  
import androidx.compose.material3.Text  
import androidx.compose.material3.TopAppBar  
import androidx.compose.runtime.Composable  
import androidx.compose.runtime.mutableStateOf  
import androidx.compose.runtime.remember  
import androidx.compose.ui.Alignment  
import androidx.compose.ui.Modifier  
import androidx.compose.ui.graphics.Color  
import androidx.compose.ui.platform.*LocalContext*import androidx.compose.ui.text.style.TextAlign  
import androidx.compose.ui.tooling.preview.Preview  
import androidx.compose.ui.unit.dp  
import androidx.compose.ui.unit.sp  
import java.util.\*  
  
class MainActivity : ComponentActivity() {  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 *setContent* **{** MainContent()  
 **}** }  
}  
  
@SuppressLint("UnusedMaterial3ScaffoldPaddingParameter")  
@OptIn(ExperimentalMaterial3Api::class)  
@Composable  
fun MainContent() {  
  
 val mContext = *LocalContext*.current  
 val mCalendar = Calendar.getInstance()  
 var selectedDay = mCalendar.get(Calendar.*DAY\_OF\_MONTH*)  
 var selectedMonth = mCalendar.get(Calendar.*MONTH*)  
  
 val mDate = remember **{** *mutableStateOf*("") **}** val resultText = remember **{** *mutableStateOf*("") **}** val mDatePickerDialog = DatePickerDialog(  
 mContext**,  
 {** \_: DatePicker**,** year: Int**,** month: Int**,** dayOfMonth: Int **->** selectedDay = dayOfMonth  
 selectedMonth = month  
 mDate.value = "$dayOfMonth/${month + **1**}/$year"  
 **},** mCalendar.get(Calendar.*YEAR*)**,** mCalendar.get(Calendar.*MONTH*)**,** mCalendar.get(Calendar.*DAY\_OF\_MONTH*)  
 )  
  
 Scaffold(  
 ) **{** Column(**modifier =** Modifier.*fillMaxSize*()**, verticalArrangement =** Arrangement.Center**, horizontalAlignment =** Alignment.CenterHorizontally) **{** Button(  
 **onClick = {** mDatePickerDialog.show()  
 **},** ) **{** Text(**text =** "Open Date Picker"**, color =** Color.White)  
 **}** Spacer(**modifier =** Modifier.*size*(**100**.*dp*))  
  
 Text(**text =** "Selected Date: ${mDate.value}"**, fontSize = 30**.*sp***, textAlign =** TextAlign.Center)  
  
 Text(  
 **text =** "Result: ${resultText.value}"**,  
 fontSize = 20**.*sp***,  
 textAlign =** TextAlign.Center**,  
 modifier =** Modifier.*padding*(**16**.*dp*)  
 )  
  
 // Добавляем кнопку для обработки действий пользователя  
 Button(  
 **onClick = {** resultText.value = *updateResultText*(selectedDay**,** selectedMonth)  
 **},  
 modifier =** Modifier.*padding*(**16**.*dp*)  
 ) **{** Text(**text =** "Process Action")  
 **}  
 }  
 }**}  
  
  
private fun updateResultText(selectedDay: Int**,** selectedMonth: Int): String {  
 val month = selectedMonth + **1** // Увеличиваем значение месяца на 1, чтобы соответствовать счету месяцев с 1 до 12  
  
 return when {  
 (month == **3** && selectedDay >= **21**) || (month == **4** && selectedDay <= **19**) -> getString(R.string.aries)  
 (month == **4** && selectedDay >= **20**) || (month == **5** && selectedDay <= **20**) -> getString(R.string.taurus)  
 (month == **5** && selectedDay >= **21**) || (month == **6** && selectedDay <= **20**) -> getString(R.string.gemini)  
 (month == **6** && selectedDay >= **21**) || (month == **7** && selectedDay <= **22**) -> getString(R.string.cancer)  
 (month == **7** && selectedDay >= **23**) || (month == **8** && selectedDay <= **22**) -> getString(R.string.leo)  
 (month == **8** && selectedDay >= **23**) || (month == **9** && selectedDay <= **22**) -> getString(R.string.virgo)  
 (month == **9** && selectedDay >= **23**) || (month == **10** && selectedDay <= **22**) -> getString(R.string.libra)  
 (month == **10** && selectedDay >= **23**) || (month == **11** && selectedDay <= **21**) -> getString(R.string.scorpio)  
 (month == **11** && selectedDay >= **22**) || (month == **12** && selectedDay <= **21**) -> getString(R.string.sagittarius)  
 (month == **12** && selectedDay >= **22**) || (month == **1** && selectedDay <= **19**) -> getString(R.string.capricorn)  
 (month == **1** && selectedDay >= **20**) || (month == **2** && selectedDay <= **18**) -> getString(R.string.aquarius)  
 (month == **2** && selectedDay >= **19**) || (month == **3** && selectedDay <= **20**) -> getString(R.string.pisces)  
 else -> getString(R.string.unknown\_sign)  
 }  
}  
  
@Preview(showBackground = true)  
@Composable  
fun DefaultPreview() {  
 MainContent()  
}

делаешь enum, его результат в mutablestate