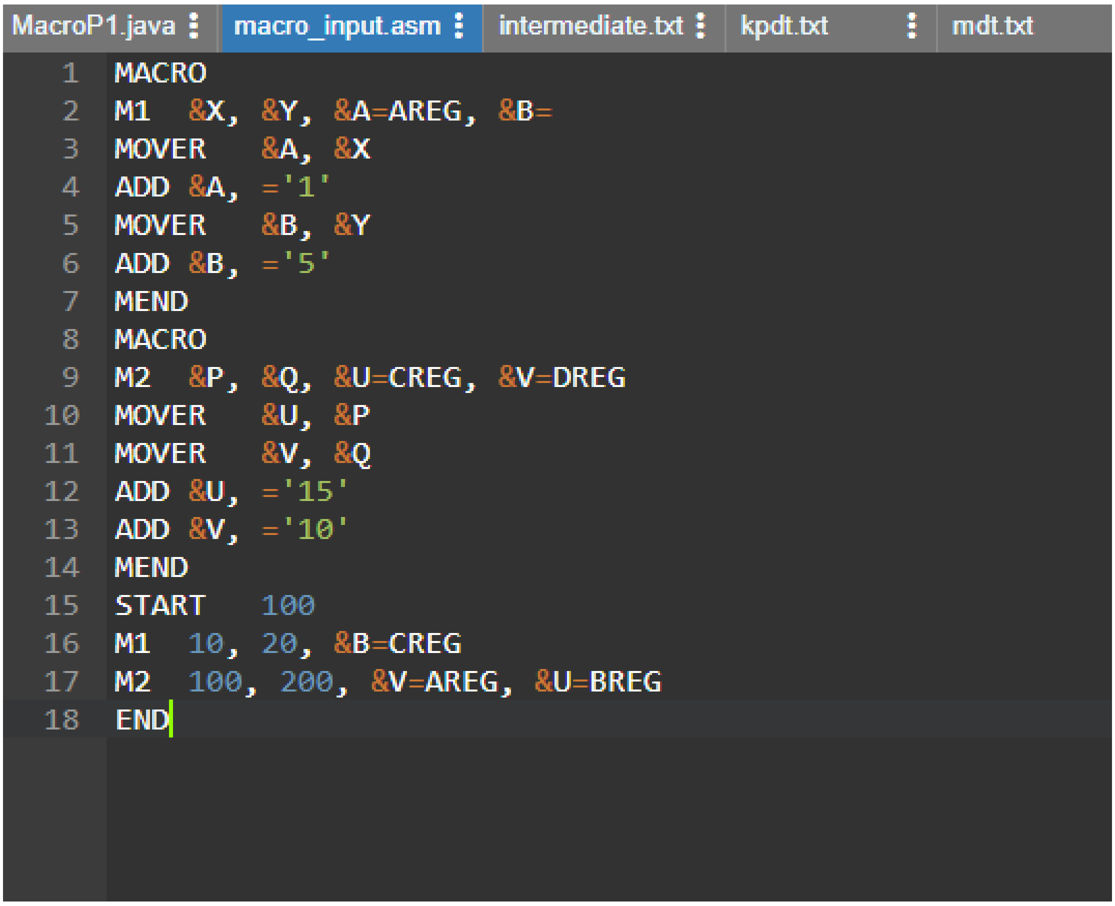
```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Iterator;
import java.util.LinkedHashMap;

public class MacroP1 {

    public static void main(String[] args) throws IOException{
        BufferedReader br=new BufferedReader(new 
FileReader("macro_input.asm"));

        FileWriter mnt=new FileWriter("mnt.txt");
        FileWriter mdt=new FileWriter("mdt.txt");
        FileWriter kpdt=new FileWriter("kpdt.txt");
        FileWriter pnt=new FileWriter("pntab.txt");
        FileWriter ir=new FileWriter("intermediate.txt");
        LinkedHashMap<String, Integer> pntab=new LinkedHashMap<>();
        String line;
        String Macroname = null;
        int mdtp=1,kpdtp=0,paramNo=1,pp=0,kp=0,flag=0;
        while((line=br.readLine())!=null)
        {

            String parts[]=line.split("\\s+");
            if(parts[0].equalsIgnoreCase("MACRO"))
            {
                flag=1;
                line=br.readLine();
                parts=line.split("\\s+");
                Macroname=parts[0];
                if(parts.length<=1)
                {
    mnt.write(parts[0]+"\t"+pp+"\t"+kp+"\t"+mdtp+"\t"+(kp==0?kpdtp:(kpdtp+1))+"\n");
                    continue;
                }
                for(int i=1;i<parts.length;i++) //processing of parameters
                {
                    parts[i]=parts[i].replaceAll("[&,]", "");
                    //System.out.println(parts[i]);
                    if(parts[i].contains("="))
                    {
                        ++kp;
                        String keywordParam[]=parts[i].split("=");
                        pntab.put(keywordParam[0], paramNo++);
                        if(keywordParam.length==2)
                        {
```

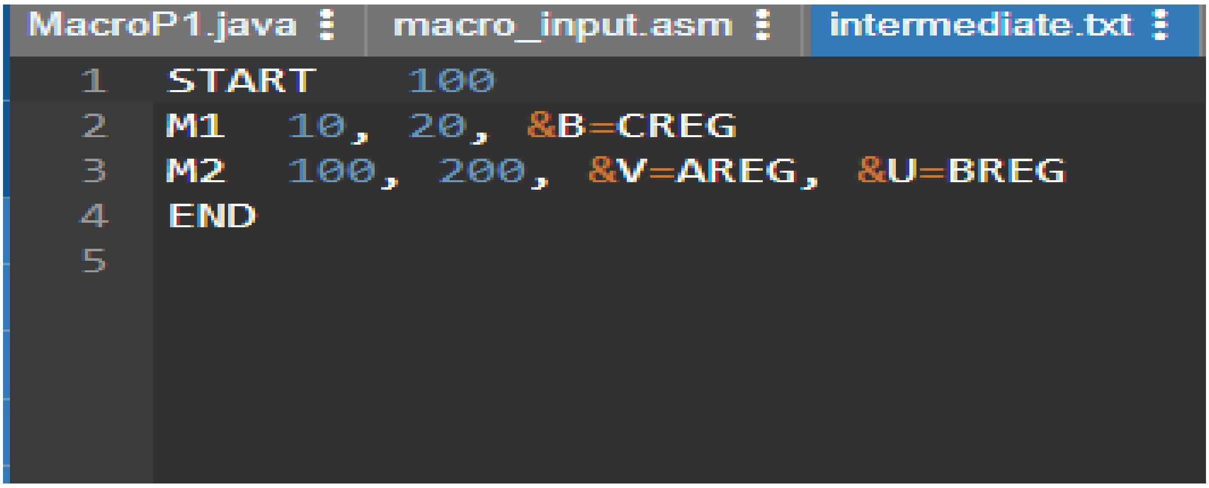
```
kpdt.write(keywordParam[0]+"\t"+keywordParam[1]+"\n");
                    }
                    else
                    {
                        kpdt.write(keywordParam[0]+"\t-\n");
                    }
                }
                else
                {
                    pntab.put(parts[i], paramNo++);
                    pp++;
                }
            }

mnt.write(parts[0]+"\t"+pp+"\t"+kp+"\t"+mdtp+"\t"+(kp==0?kpdtp:(kpdtp+1))+"\n");
            kpdtp=kpdtp+kp;
            //System.out.println("KP="+kp);


        }
        else if(parts[0].equalsIgnoreCase("MEND"))
        {
            mdt.write(line+"\n");
            flag=kp=pp=0;
            mdtp++;
            paramNo=1;
            pnt.write(Macroname+":\t");
            Iterator<String> itr=pntab.keySet().iterator();
            while(itr.hasNext())
            {
                pnt.write(itr.next()+"\t");
            }
            pnt.write("\n");
            pntab.clear();
        }
        else if(flag==1)
        {
            for(int i=0;i<parts.length;i++)
            {
                if(parts[i].contains("&"))
                {
                    parts[i]=parts[i].replaceAll("[&,]", "");
                    mdt.write("(P,"+pntab.get(parts[i])+")\t");
                }
                else
                {
                    mdt.write(parts[i]+"\t");
                }
            }
```

```
                        mdt.write("\n");
                        mdtp++;
                }
                else
                {
                        ir.write(line+"\n");
                }
        }
        br.close();
        mdt.close();
        mnt.close();
        ir.close();
        pnt.close();
        kpdt.close();
        System.out.println("MAcro PAss1 Processing done. :)");
    }

}
```

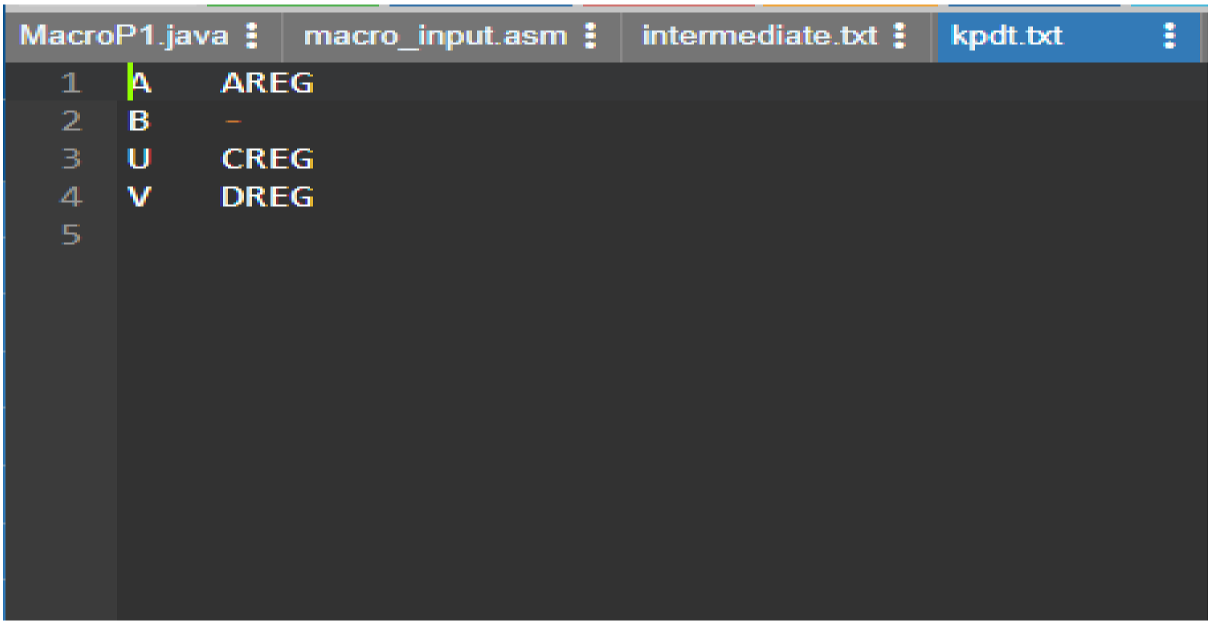
Output:



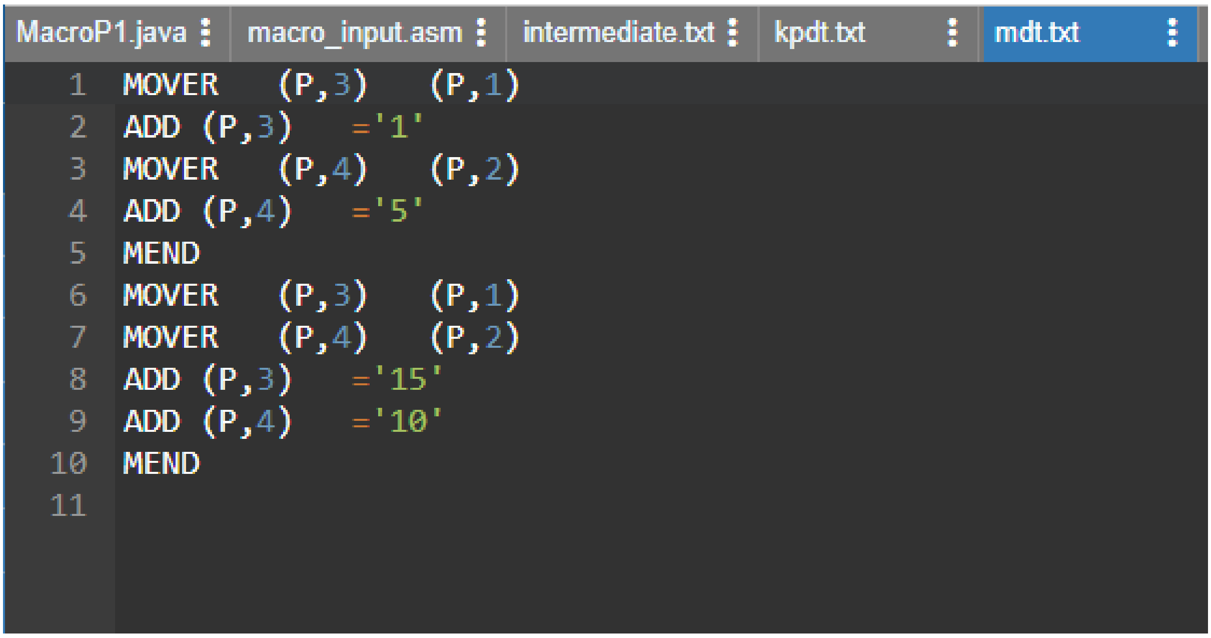
```
MACRO
M1  &X, &Y, &A=AREG, &B=
MOVER   &A, &X
ADD &A, ='1'
MOVER   &B, &Y
ADD &B, ='5'
MEND
MACRO
M2  &P, &Q, &U=CREG, &V=DREG
MOVER   &U, &P
MOVER   &V, &Q
ADD &U, ='15'
ADD &V, ='10'
MEND
START   100
M1  10, 20, &B=CREG
M2  100, 200, &V=AREG, &U=BREG
END
```



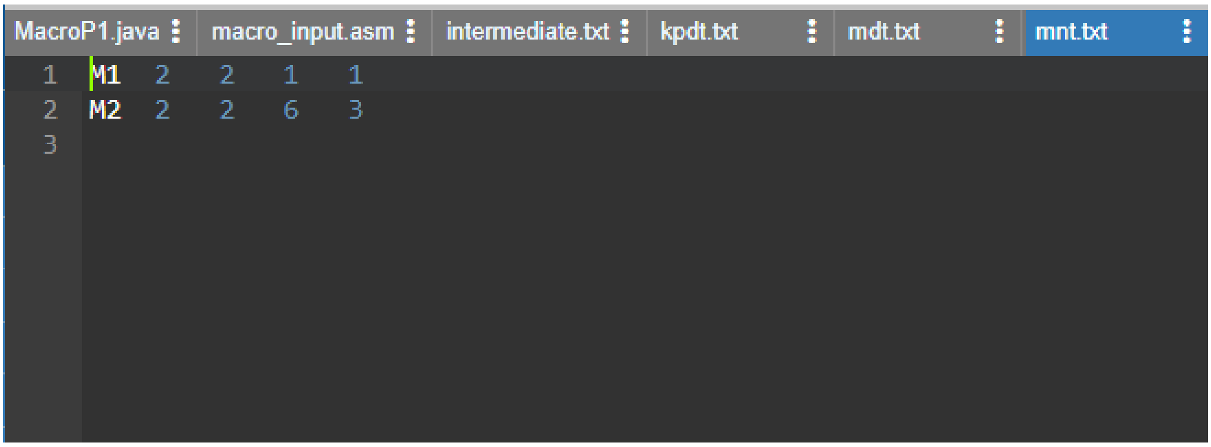
```
START    100
M1   10, 20, &B=CREG
M2   100, 200, &V=AREG, &U=BREG
END
```



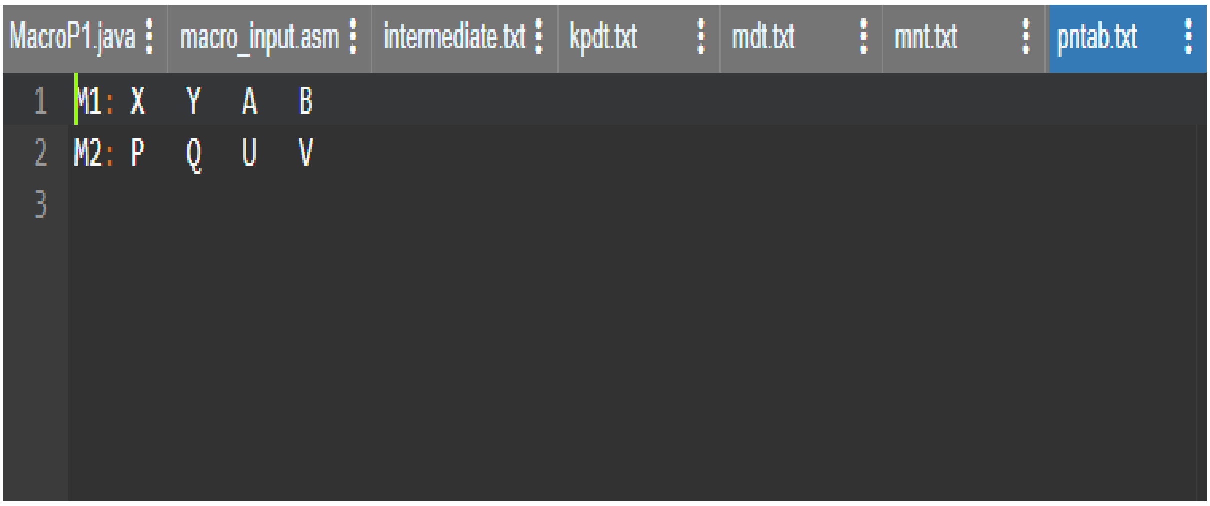
```
A     AREG
B     -
U     CREG
V     DREG
```



```
MOVER   (P,3)   (P,1)
ADD (P,3)   ='1'
MOVER   (P,4)   (P,2)
ADD (P,4)   ='5'
MEND
MOVER   (P,3)   (P,1)
MOVER   (P,4)   (P,2)
ADD (P,3)   ='15'
ADD (P,4)   ='10'
MEND
```



```
M1   2   2   1   1
M2   2   2   6   3
```



```
M1: X   Y   A   B
M2: P   Q   U   V
```