Operating System Design

Design equivalent c program to execute corresponding commands to carry out related task.

Interpreter

```
#include<stdio.h>   #include<string.h>
main( )
{  char x[20]; int a[200];
   do{scanf("%s",x);
        if((x[1]=='=')&&(x[3]=='+')) a[x[0]]=a[x[2]]+a[x[4]];
        if((x[1]=='=')&&(x[2]<60)) a[x[0]]=x[2]-48;
        if(x[1]=='r') {  printf("%d\n",a[x[6]]);sleep(1);  }
    } while(1);
}
```

1. Command: k=7 m=2 u=k+m print(u)
   Outputs   : 9
   (A) Implement copy
   Command: k=7 u=k print(u)
   Output: 7
2. Implement simple conditional
   Command: c=3 u=5 k=7 g=8 if(u>k)g=c print(g)
   Output   : 8
   when u=9 in previous command, o/p: 3
3. Implement indirect (source)
   Command: c=9 e=6 f=9 k=[5] print(k)
   Output   : 6
4. Implement indirect (destination)
   Command: c=5 d=6 [4]=8 print(d)
   Output   : 8
   and print(c) outputs 5
5. Modify above
   Command: g=4 d=7 e=9 k=[g]   print(k)
   Output   : 7
   Command: g=4 d=7 e=9 [g]=8 print(d)
   Output   : 8
6. Conditional of one statement
   Command: if(u>k)a=b+c. In place of a=b, there can be any assignment statement like a=5, a=b+c or a=5+c

Two processes (id z=1 and 2)

```
char x[20]; int a[200],b[200];a[122]=1;b[122]=2;
do{scanf("%s",x);
     if((x[1]=='=')&&(x[3]=='+'))  { a[x[0]]=a[x[2]]+a[x[4]]; b[x[0]]=b[x[2]]+b[x[4]]; }
     if((x[1]=='=')&&(x[2]<60)) {  a[x[0]]=x[2]-48;  b[x[0]]=x[2]-48;  }
     if(x[1]=='r') {  printf("%d %d\n",a[x[6]],b[x[6]]);sleep(1);  }
 } while(1);
```

1. Command: k=7 m=k+z print(m)
   Output   : 8 9
   Command: k=z+z k=z+k k=k+k print(k)
   Output   : 6 12

Multi processor (process id's z=1..n)

```
char x[20]; int a[50][200],i,n; printf("Give number of processes");scanf("%d",&n);
for(i=1;i<=n;i++){ a[i][122]=i; a[i][118]=0;}
do{scanf("%s",x);
     for(i=1;i<=n;i++)
     {   if((x[1]=='=')&&(x[3]=='+')) a[i][x[0]]=a[i][x[2]]+a[i][x[4]];
         if((x[1]=='=')&&(x[2]<60)) a[i][x[0]]=x[2]-48;
         if(x[1]=='r') {  printf("%d\n",a[i][x[6]]);sleep(1);  }
     }
 } while(1);
```

1. Command: k=7 g=z+z g=g+g m=k+g print(m) (4 processes)
   Output   : 11 15 19 23
2. process creation using fork

   ```
   if(x[1]=='o'){
         m=n;
         for(i=1;i<=m;i++){
                   n++;
                   for(j=0;j<199;j++)
                             a[n][j]=a[i][j];
                   a[n][122]=n;
             }
      }
   ```

   Command: k=2 print(k) fork k=k+z print(k) fork print(k) k=k+z print(k) (n=1)
   Output   :2 34 3434 4668
3. Implement fork with return value. In parent the id of child is returned. In child 0 is returned.
   Command: g=fork u=z+z u=u+u u=u+g print(u) (n=1 o/p68) (n=2 o/p 7 12 12 16)
   Output   : (for n=1) 6 8, (for n=2) 7 12 12 16

4. Let ‘v’ store father’s id.
   Command: a=fork print(a) print(v) b=fork print(b) print(v)
   Output : 20 01 3400 0112

//This for 2nd case fork and parent child connection

```
if(x[3]==‘o’){
        …
        a[i][x[0]]=n;
        a[n][x[0]]=0;
        a[n][118]=i;
}
```