# APPENDIX B

```
/* The Parallel Minimum Spanning Tree */
main mst

deflog  (TRUE, 1);
deflog  (FALSE, 0);

char parallel tail[$], head[$];
int  parallel weight[$], state[$];
char scalar node;
index parallel xx[$];
logical parallel nxtnod[$],graph[$],result[$];


associate head[$],tail[$],weight[$],state[$] with graph[$];

read tail[$] head[$] weight[$] in graph[$];

setscope graph[$]
node = tail[mindex(weight[$])];
endsetscope;

if(node .eq. tail[$]) then state[$] = 2; else state[$] =3; endif;
while xx in (state[$] .eq. 2)
asmcode
   printf("node = %c\n",NODE);
endasm;
if(state[$] .eq. 2)then nxtnod[$] = mindex(weight[$]); endif;
node = head[nxtnod[$]];
state[nxtnod[$]]=1;
if(head[$] .eq. node .and. state[$] .ne. 1) then
  state[$] = 0;
endif;
if(state[$] .eq. 3 .and. node .eq. tail[$]) then
  state[$] = 2;
endif;
nxtnod[$] = FALSE; /* must clear when done for next iteration */
endwhile xx;

if (state[$] .eq. 1) then result[$]= TRUE ; endif;
print tail[$] head[$] weight[$] in result[$];


end;
```