Problem SA1 (10 points)

```
/* Assume this program is spawned with only fds 0,1 and 2 open and
 * the signal handling table at default values.
 * System pipe buffer size is 64K */
main()
{
  char buf[8192];
  int a[2],i;
  int status;
          pipe(a);  /* Assume this syscall succeeds */
          switch(fork())  /*        ditto                */
          {
            case 0:
                    for(i=0;i<65536;i++)
                              write(a[1],buf,8192);
                    return 0;
          }
          if (wait(&status)== -1)
                    perror("OMFG Wait failed!!!!");
          else
                    fprintf(stderr,"Child returned %d\n",status);
          return 0;
}
```

What does this program print? **Explain your answer** for full credit!

Nobody ever reads from the read side of the pipe. The pipe has a capacity of 64K as stated but the child tries to write 512K. (There is no SIGPIPE or EPIPE because the read side of the pipe is still open -- in both processes actually). The write system call blocks and the child never exits. The parent is stuck in the wait system call and never gets out of it since the child is stuck. Nothing prints.

Code Fragment MC#1

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/signal.h>
#include <setjmp.h>

jmp_buf jb;
int counter;

void handler(int sn)
{
         counter++;
         longjmp(jb,1);
}

main()
{
 struct sigaction sa;
 int ppid,i;
         sa.sa_handler=handler;
         sigemptyset(&sa.sa_mask);
         sa.sa_flags=0;
         sigaction(SIGINT,&sa,NULL);
         if (setjmp(jb)!=0)
         {
                  fprintf(stderr,"counter = %d\n",counter);
                  if (counter>=1000)
                  {
                           fprintf(stderr,"All done\n");
                           exit(0);
                  }
                  goto GOTO_IS_EVIL;
         }
         ppid=getpid();
         switch(fork())
         {
           case 0:
                  for(i=0;i<1000;i++)
                           kill(ppid,SIGINT);
                  exit(0);
         }
 GOTO_IS_EVIL:
         for(;;)
                  ;
}
```