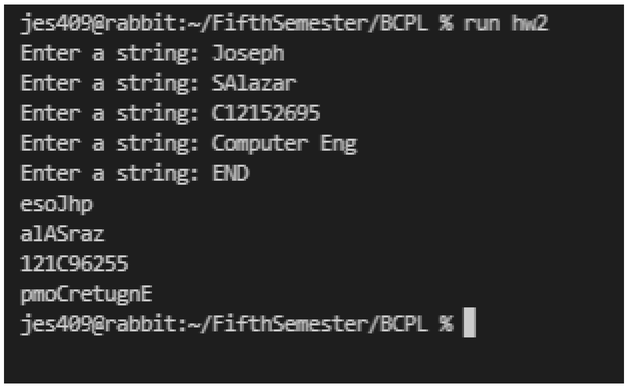
Joseph Salazar HW #2 Murrell ECE 421

```
import "io"

manifest
{
   node_str =0,
   node_nxt = 1,
   sizeof_node = 2
}

let new_node(x) be
{
   let p = newvec(sizeof_node);
   p ! node_str := x;
   p ! node_nxt := nil;
   resultis p;
}

let insert(node, str) be
{
   switchon node into
   {
      case nil:
         resultis new_node(str);
      endcase;
      default:
         node ! node_nxt := insert(node ! node_nxt, str);
   }
```

```
    resultis node;
}

let printlist(head) be
{
    if head = nil then return;
    out("%s\n", head ! node_str);
    printlist(head ! node_nxt);
}

let mystrdup(in, length, bytes) be
{
    let str;
    bytes := (length *4) + bytes;
    str := newvec(bytes);
    for i =0 to bytes -1 do
    {
        byte i of str := byte i of in;
    }
    byte bytes of str := nil;

    resultis str;
}

let string_in() be
{
    let ch, str;        //char and string entered
    let strlen, bytes;
    let temp = vec(80);
    let nxt;

    ch:=inch();
    for i = 0 to 79 do
    {
        for j = 0 to 3 do
        {
            switchon ch into
            {
                case '\n':
                    strlen := i;
                    bytes := j;
                    i := 80; j :=4;
                    endcase;
                default:
                    temp ! i:= ((temp ! i) << 8) + ch;
                    ch := inch();
            }
        }
    }
    str := mystrdup(temp, strlen, bytes);
    resultis str;
}

let compare(in, end) be
{
    let len1, len2, c1, c2, diff, boolean;
    let i = 0;
    len1 := strlen(in);

    len2 := strlen(end);

    boolean := 1; //the same

    diff := len1 - len2;


    if diff /= 0 then

        boolean := 0;

        resultis boolean;


    while true do

    {

        let c1 = byte i of in; let c2 = byte i of end;

        if c1 /= c2 then

            boolean := 0;

        i := i +1;

        if i >= len1 then

            break;

        if i >= len2 then

            break;

    }

    resultis boolean;

    // 1 is the same

    // 0 is different

}



let start() be

{

    let str;
    let flag;
    let links = nil;
    let heap = vec(10000);
    init(heap, 10000);

    for i =0 to 9 do
    {
        out("Enter a string: ");
        str:= string_in();
        flag:=compare(str, "DNE");
        switchon flag into
        {
            case 1:
                break;
            endcase;
            default:
                links := insert(links,str);
        }
    }
    printlist(links);
    freevec(links);
}
```



```
jes409@rabbit:~/FifthSemester/BCPL % run hw2
Enter a string: Joseph
Enter a string: SAlazar
Enter a string: C12152695
Enter a string: Computer Eng
Enter a string: END
esoJhp
alASraz
121C96255
pmoCretugnE
jes409@rabbit:~/FifthSemester/BCPL %
```