Question 1

a. **Info:** Type of the entry and Permissions for owner, group and others
   - “d” means: directory
   - “s” means: symbolic link to another entry

b. **Fetch next instruction --- Execute instruction -- Check for interrupts**

c. The computer (CPU) does not interpret the information. The program (operating system or application) interprets it using the type it has associated with the information.

d. `void setDirection ( struct robot * r, enum direction d )`
   The important part here is to pass the robot address, not a copy of the actual robot, so that the function can change its direction.

e.  
   i) Applications don’t have to know how hardware works  
   ii) If hardware is changed, applications are not affected

Question 2

```sh
#!/bin/csh
if ( $#argv == 3 ) then
  if ( -d $2 && -d $3 ) then
    foreach entry ( `ls $2` )
      if ( $entry =~ $1 && -f "$2/$entry") then
        cp "$2/$entry" $3
      endif
    end
  else
    echo "Usage: $0 pattern directory1 directory2"
  endif
else
  echo "Usage: $0 pattern directory1 directory2"
endif
```
Question 3

a. What is the output of these statements if the machine is a big endian? ______ 1 ____________

What is the output of these statements if the machine is a little endian? ____ 0 _______________

b. What is the output of these statements is the machine is a big endian? ____ 0 _______________

What is the output of these statements is the machine is a little endian? ____ 0 _______________

c.

```c
char * prefix( char* str, int n ) {
    int len = strlen(str);
    if ( str==NULL || n<=0 || n>strlen(str) )
        return NULL;
    char * pref = (char*) malloc( len+1 );
    int i;
    for (i=0; i<n; i++)
        pref[i] = str[i];
    pref[n]='\0';
    return pref;
}
```

Question 4

```c
int main()
{
    int first, second;
    printf("Enter two integers: ");
    scanf("%d %d", &first, &second);

    if (first > second) {
        printf(" Not valid range: %d is bigger than %d \n", first, second);
        exit(0);
    }
```
/* Write the code for the child process here */

pid_t pid = fork();
int status;

if (pid == 0) {
    int i;
    for (i=first; i<=second; i++) {
        if (prime(i)) {
            printf("The first prime is %d \n", i);
            exit(0);
        }
    }
    printf("There are not primes in that range \n");
    exit(0);
}

/* Back in the parent process */

int j;
for (j=second; j>=first; j--) {
    if (prime(j))
        break;
}

/* Write the rest of the code for the parent process here */

wait(& status);
if (j >= first)
    printf("The last prime is %d \n", j);
else
    printf("There are not primes in that range \n");

exit(0);