Pseudocode

```
# model: pretrained (e.g. distilled) model
# layer_norm: layer normalization layer
# cls_head: linear classifier -> nn.Linear(D, C)
# x: batch of images (B, 3, H, W)
def image_downstream_forward(model, layer_norm, cls_head, x, linear_probe):

    if linear_probe:
      with torch.no_grad():
        x = model(x) # (B, T, D)
    else:
      x = model(x) # (B, T, D)

    x = x[:, 1:] # remove cls token (B, T-1, D)
    x = x.mean(dim=1) # mean over all patches (B, D)
    x = layer_norm(x)
    x = cls_head(x) # (B, C)
    pred = x.argmax(dim=-1) # (B, )
    return pred
```

Code 1: Pytorch pseudocode for the forward pass during finetuning or linear probing of a pretrained model on an image classification tasks. The output of the forward pass is the predicted class index for each image in the batch.

```
# model: pretrained (e.g. distilled) model
# bert_pooler: pretrained BERT pooler layer with tanh activation
# dropout: dropout layer
# cls_head: roberta classification head
# x: batch text tokens (B, M+2) -> M: max sequence length, 2: cls and sep token
# target: batch of labels (B, )
# regression: boolean flag for regression tasks
# metric: either Accuracy, F1, or Spearman correlation
def glue_forward(model, bert_pooler, dropout, cls_head,
                 x, target, regression, metric):

    x = model(x) # (B, M+2, D)

    x = x[:, 0] # take cls token (B, D)
    x = dropout(bert_pooler(x)) # (B, D)
    x = cls_head(x) # (B, C) -> C: number of classes

    if regression:
      x = x.squeeze() # (B, ) -> remove the last dimension, as C=1
      pred = x
      loss = mse_loss(x, target)
    else:
      pred = x.argmax(dim=-1) # (B, )
      loss = cross_entropy(x, target)

    score = metric(pred, target)

    return loss, pred, score
```

Code 2: Pytorch pseudocode for the forward pass during finetuning on GLUE benchmark tasks.

```
# teacher_model: ResNet-50-A1 model
# image_encoder: Image encoder of the multimodal student model
# text_encoder: Text encoder of the multimodal student model
# shared_encoder: Shared encoder of the multimodal student model
# imgs: batch of images (B, 3, H, W)
# captions: batch of image captions (B, 64)
# kl_div: KL-Divergence
# clip_loss: Contrastive loss used in CLIP
def forward(teacher_model, image_encoder, text_encoder,
            shared_encoder, imgs, captions):

    with torch.no_grad():
      target = teacher_model(imgs) # (B, 1000)

    img_layer_res = shared_encoder(image_encoder(imgs)[:, 0])
    # [(B, 3072), (B, 768), (B, 1000)]

    text_layer_res = shared_encoder(text_encoder(captions)[:, 0])
    # [(B, 3072), (B, 768), (B, 1000)]

    kl_loss = 1/2*kl_div(target, img_layer_res[2]) +
              1/2*kl_div(target, text_layer_res[2])

    itc_loss = 1/3*clip_loss(img_layer_res[0], text_layer_res[0]) +
               1/3*clip_loss(img_layer_res[1], text_layer_res[1]) +
               1/3*clip_loss(img_layer_res[2], text_layer_res[2])

    loss = kl_loss + itc_loss

    return loss
```

Code 3: Abstract code used in the forward pass for distilling the multimodal Transformer SHRe from a pretrained ResNet-50-A1 model.

```
# image_vq: Pretraing image vector quantizer
# image_encoder: Image encoder of the multimodal student model (Data2Vec2)
# text_encoder: Text encoder of the multimodal student model (BERT)
# shared_encoder: Shared Transformer layer of the multimodal student model
# cls_head: Classification head of the multimodal student model
# imgs: batch of images (B, 3, H, W)
# captions: batch of image captions (B, 64)
# clip_loss: Contrastive loss used in CLIP
def forward(image_vq, image_encoder, text_encoder,
            shared_encoder, cls_head, imgs, captions):

    with torch.no_grad():
      target = image_vq.quantize_image(imgs) # (B,)

    img_layer_res = shared_encoder(image_encoder(imgs))
    # [(B, 3072), (B, 768)]

    image_pred = cls_head(img_layer_res[1]) # (B, 1024)

    text_layer_res = shared_encoder(text_encoder(captions))
    # [(B, 3072), (B, 768)]

    text_pred = cls_head(text_layer_res[1]) # (B, 1024)

    kd_loss = 1/2*cross_entropy(target, image_pred) +
              1/2*cross_entropy(target, text_pred)

    itc_loss = 1/2*clip_loss(img_layer_res[0], text_layer_res[0]) +
               1/2*clip_loss(img_layer_res[1], text_layer_res[1])

    loss = kd_loss + itc_loss

    return loss
```

Code 4: Pytorch pseudocode used in the forward pass for distilling S-SMKE with quantized image representations. The vector quantizer returns the index m of the closest codebook vector $\boldsymbol{q}_m$ for each image in the batch.