LAMPIRAN

Lampiran A. Media Pembelajaran

Kuncup tumbuhan menangkap dan menahan air, yang harus disaring untuk memisahkan serangga dan potongan kecil daun-daun.

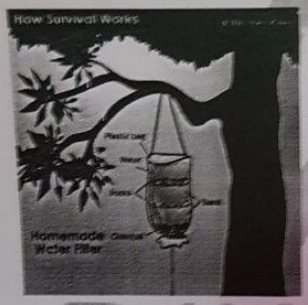
Air yang kita dapatkan di alam, kadang kala bisa kita minum secara langsung bisa juga tidak. Jenis air yang dapat langsung kita minum adalah air tawar, embun, air hujan dan mata air. Sedangkan air rawa, sungai besar, air tergenang harus kita murnikan dahulu sebelum dikonsumsi.

Cara memurnikan air :

Merebusnya terlebih dahulu

Memberinya serbuk atau tablet penjernih air (Tawas/Kaporit)

Penyulingan (Krikil/Pasir/Ijuk)



c. API

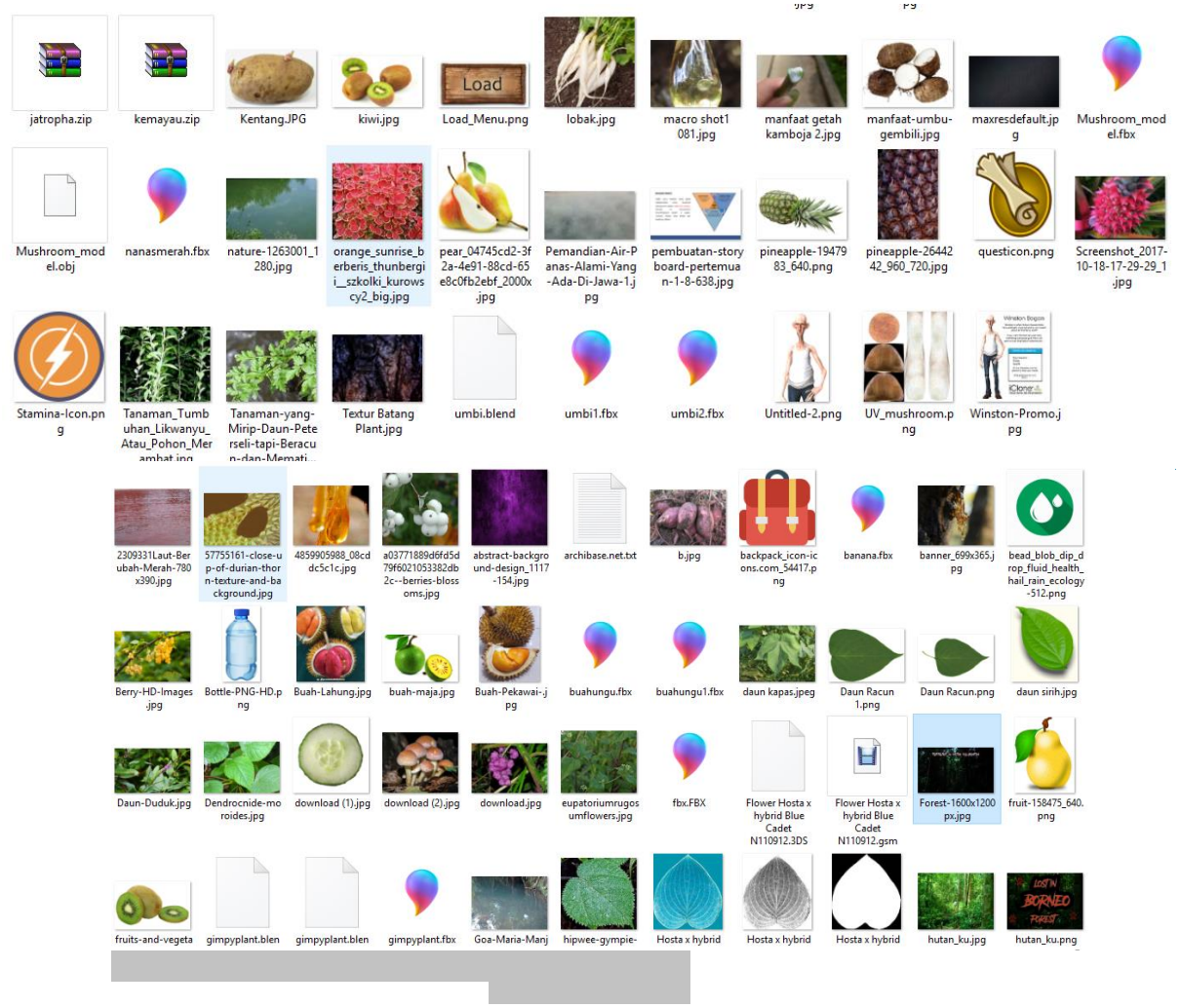
Didalam situasi survival, kemampuan untuk menyalakan api dapat membuat perbedaan antara hidup dan sekarat. Api dapat memenuhi banyak banyak kebutuhan. Dapat menyediakan kehangatan dan kenyamananr.Juga untuk memasak dan menghangatkan makanan,serta juga dengan makanan yang hangat dapat membuat kita bisa menghemat kalori dalam tubuh kita yang biasanya diproduksi sewaktu tubuh memproduksi panas tubuh. Anda dapat menggunakan api untuk memurnikan air, mensterilkan perban, isyarat untuk penolong, dan memberikan perlindungan dari binatang. Dan secara pisikologi memberikan kedamaian pikiran dari ketegangan serta persahabatan. anda dapat juga menggunakan api untuk menghasilkan perkakas dan senjata.

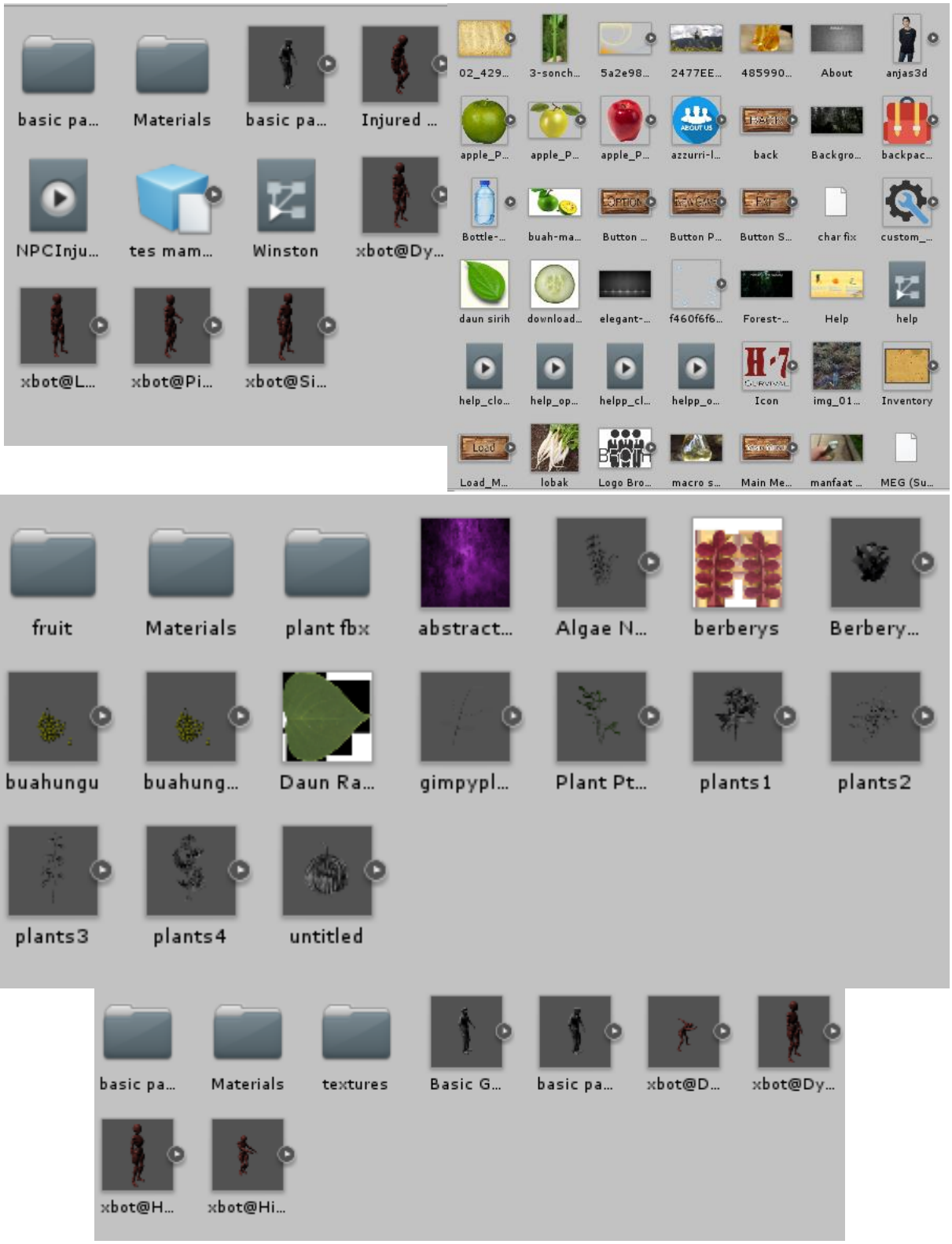
Api dapat menyebabkan permasalahan, juga yaitu dapat menyebabkan kebakaran hutan. Api dapat juga menyebabkan membakar peracunan karbon monoksida ketika dinyalakan dalam shelter perlindungan.

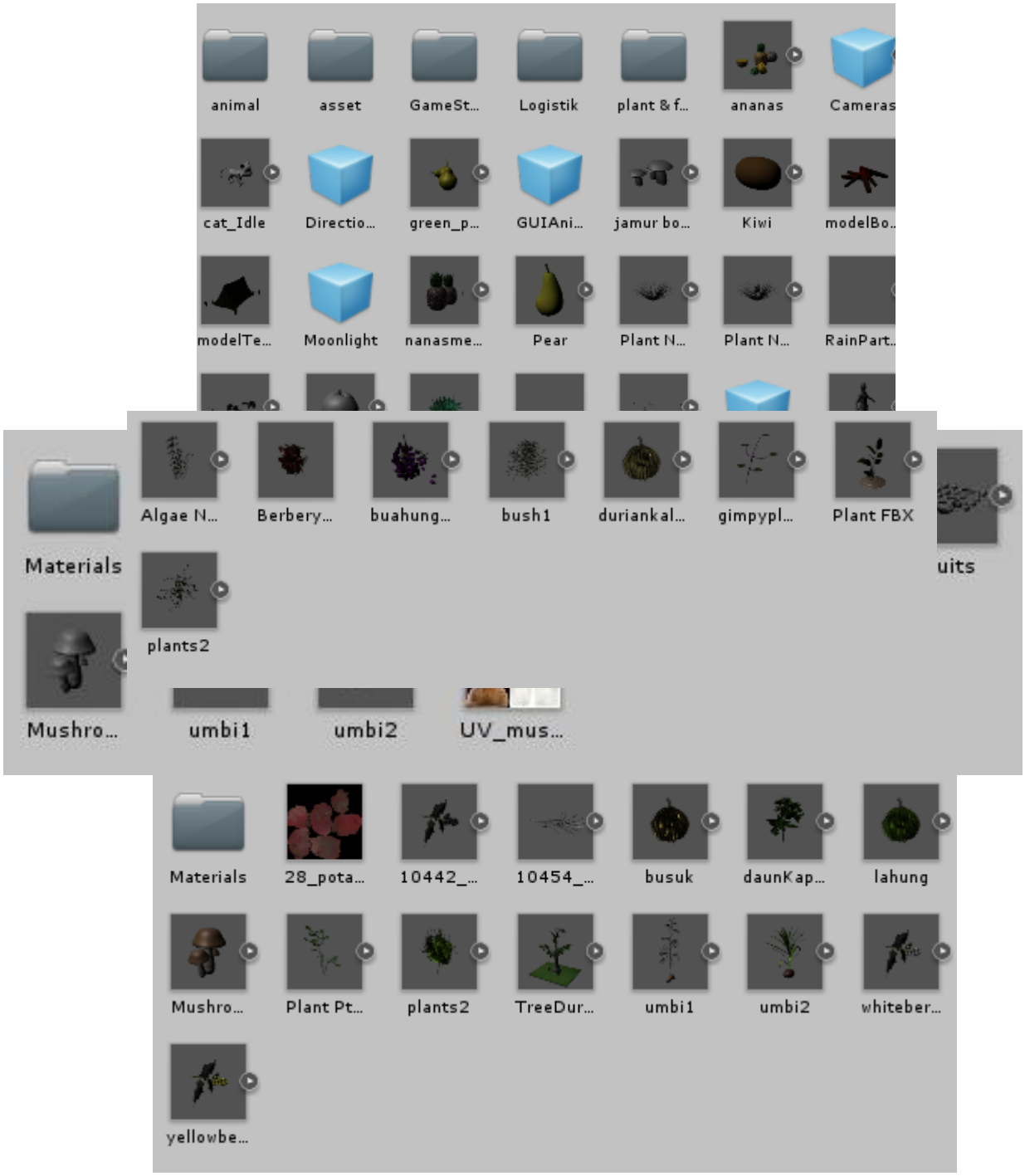
PRINSIP DASAR API

KORPS INSTRUKTUR MAPALA UMY | DIKTAT DIKSAR XXVIII 59

Lampiran B. Pengumpulan Bahan




basic pa...
Materials
basic pa...
Injured ...
NPCInju...
tes mam...
Winston
xbot@Dy...
xbot@L...
xbot@Pi...
xbot@Si...
02_429...
3-sonch...
5a2e98...
2477EE...
485990...
About
anjas3d
apple_P...
apple_P...
apple_P...
azzurri-l...
back
Backgro...
backpac...
Bottle-...
buah-ma...
Button ...
Button P...
Button S...
char fix
custom_...
daun sirih
download...
elegant-...
f460f6f6...
Forest-...
Help
help
help_clo...
help_op...
helpp_cl...
helpp_o...
Icon
img_01...
Inventory
Load_M...
lobak
Logo Bro...
macro s...
Main Me...
manfaat ...
MEG (Su...
fruit
Materials
plant fbx
abstract...
Algae N...
berberys
Berbery...
buahungu
buahung...
Daun Ra...
gimpypl...
Plant Pt...
plants1
plants2
plants3
plants4
untitled
basic pa...
Materials
textures
Basic G...
basic pa...
xbot@D...
xbot@Dy...
xbot@H...
xbot@Hi...


animal
asset
GameSt...
Logistik
plant & f...
ananas
Cameras
cat_Idle
Directio...
green_p...
GUIAni...
jamur bo...
Kiwi
modelBo...
modelTe...
Moonlight
nanasme...
Pear
Plant N...
Plant N...
RainPart...
Materials
Algae N...
Berbery...
buahung...
bush1
duriankal...
gimpypl...
Plant FBX
uits
plants2
Mushro...
umbi1
umbi2
UV_mus...
Materials
28_pota...
10442_...
10454_...
busuk
daunKap...
lahung
Mushro...
Plant Pt...
plants2
TreeDur...
umbi1
umbi2
whiteber...
yellowbe...

Lampiran C. Pengkodean *Game*

1. Class Change*Scene:* Untuk berpindah – pindah *scene* dan menyembunyikan dan memunculkan objek.

```
public class Change Scene : MonoBehaviour {
  // Beberapa contoh fungsi pergantian Scene, memunculkankan dan menghilangkan objek
  // Use this for initialization
  void Start () {
    options = GameObject.FindWithTag("setting");

    options.SetActive(false);
    help.SetActive(false);
    x = false;
    Cursor.visible = false;
    Cursor.lockState = CursorLockMode.Locked;
    ImageCanv.SetActive(false);
      }
      // Update is called once per frame
      void Update () {

    if (Input.GetKey(KeyCode.Escape))
    {
      if (Cursor.visible == false)
      {

        Cursor.visible = true;
        Cursor.lockState = CursorLockMode.Confined;
        camera.enabled = false;
        camera1.enabled = false;
      }
      else
      {
        Cursor.visible = false;
        Cursor.lockState = CursorLockMode.Locked;
        camera.enabled = true;
        camera1.enabled = true;
      }
    }

    if (Input.GetKey(KeyCode.LeftControl))
    {
      Cursor.lockState = CursorLockMode.None;
      if (Cursor.visible == false)
      {
```

```
            Cursor.visible = true;
            Cursor.lockState = CursorLockMode.Confined;
            camera.enabled = false;
            camera1.enabled = false;
        }
        else if (Cursor.visible == true)
        {
            Cursor.visible = false;
            Cursor.lockState = CursorLockMode.Locked;
            camera.enabled = true;
            camera1.enabled = true;
        }
    }

    if (Input.GetKey(KeyCode.H))
    {
        help.SetActive(true);
        Cursor.visible = true;
    }
        }
public void LoadOption Scene()
{
    Scene Manager.Load Scene(2);
}

public void LoadMainMenu Scene()
{
    Scene Manager.Load Scene(0);
}

public void LoadGampe Scene()
{
    Scene Manager.Load Scene(7);
}

public void exit Game()
{
    Application.Quit();
}
public void loadAbout()
{
    Scene Manager.Load Scene(5);
}
public void LoadOpening()
{
```

```
    SceneManager.LoadScene(6);
}
public void LoadOption()
{

    if(x == false)
    {
        options.SetActive(true);
        x = true;
    }
    else
    {
        options.SetActive(false);
        x = false;
    }
}
public void LoadHelp()
{
    help.SetActive(true);
    Cursor.visible = true;
}
public void HideHelp()
{
    help.SetActive(false);
}

public void ShowBag()
{
    animBag.SetBool("isOpen", true);
}

public void HideBag()
{
    animBag.SetBool("isOpen", false);
}
public void Dancing()
{

    if (x == false)
    {
        animDance.SetBool("Dance", true);
        x = true;
    }
    else
    {
        animDance.SetBool("Dance", false);
```

```
            x = false;
        }
    }
}
```

2. Class AutoMove: Untuk menyeleksi apakah logistic yang dibawa tidak mengandung racun.

```
public class AutoMove : MonoBehaviour {

    // Clas untuk mengoperasikan fungsi pemeriksaan saat pemain lapor ke npc
    void Update()
    {
        if (x == true)
        {
            y -= Time.deltaTime / 1;
            if(y <= 0)
            {
                animtextnotif.SetBool("isOpen", false);
                y = 4;
                x = false;
            }
        }
    }
    void OnCollisionEnter()
    {
        if (cc.questmissText.text == "Cari air minum (2/2)")
        {
            if(cc.questm_1done >= 2)
            {
                SceneManager.LoadScene(8);
            }
            else
            {
                notifText.text = "Air dalam botol mengandung racun, buang dan carilah
air bersih lagi!";
                animtextnotif.SetBool("isOpen", true);
                x = true;
            }
        }
        else if(cc.questmissText.text == "Cari 2 jenis makanan (2/2)")
        {
            if(cc.questDone >= 2)
            {
                SceneManager.LoadScene(9);
            }
            else
            {
```

```
            notifText.text = "Tumbuhan yang diambil ada yang mengandung racun, buang dan carilah yang tidak beracun!";
            animtextnotif.SetBool("isOpen", true);
            x = true;
        }
    }
    else if (cc.questmissText.text == "Cari tumbuhan tidak beracun dengan tes (1/1)")
    {
        if(cc.questm_3completed >= 1)
        {
            SceneManager.LoadScene(10);
        }
        else
        {
            notifText.text = "Tumbuhan yang diambil ada yang mengandung racun, buang dan carilah yang tidak beracun!";
            animtextnotif.SetBool("isOpen", true);
            x = true;
        }
    }
    else if (cc.questmissText.text == "Cari 3 jenis makanan dengan semua metode (3/3)")
    {
        if (cc.questm_4done >= 3)
        {
            SceneManager.LoadScene(4);
        }
        else
        {
            notifText.text = "Tumbuhan yang diambil ada yang mengandung racun, buang dan carilah yang tidak beracun!";
            animtextnotif.SetBool("isOpen", true);
            x = true;
        }
    }

  }
}
```

3. Class CosumtionControll: Untuk memberikan *feedback* hasil pada logistic yang telah diuji dan hasil interaksi dengan logistik.

```
public class ConsumptionControll : MonoBehaviour {

  // Class untuk mengoperasikan fungsi makan, ambil, tes dan lihat gambar
```

```
    // Use this for initialization
    void Start () {
        btn.GetComponent<Image>().enabled = false;
        notifWall.SetActive(false);
    }
    // Update is called once per frame
    void Update()
    {
        Vector3 fwd = transform.TransformDirection(Vector3.forward);
        if (Physics.Raycast(transform.position, fwd, out hit, raycastLeght, layerMaskNew.value))
        {
            if (hit.collider.CompareTag("Consumable"))
            {
                crossHairActive();
                raycastedObj = hit.collider.gameObject;

                itemName = raycastedObj.GetComponent<ItemsProperties>().itemName;
            }
            if (hit.collider.CompareTag("wall"))
            {
                raycastedObj = hit.collider.gameObject;
                notifWall.SetActive(true);
            }
        }
        else
        {
```

```
    crossHairNormal();
    notifWall.SetActive(false);
 }
foreach (Transform child in invenPanel.transform)
 {
    // ketika item sudah ada di inven
    if (child.gameObject.tag == "good water")
    {
        string c = child.Find("TextICount").GetComponent<Text>().text;
        int icount = System.Int32.Parse(c);
        if (icount >= 2)
        {
            btn.GetComponent<Image>().enabled = true;
            questmissText.text = "Cari air minum (2/2)";
            questmiss1Text.text = "Bawa ke tempat Kakek Legenda";
            questmiss1Text.color = Color.yellow;
            btn.GetComponent<Image>().sprite = textureop;
            questm_1done = 2;
        }
        else if (icount <= 1)
        {
            questmissText.text = "Cari air minum (1/2)";
            questmissText.color = Color.yellow;
            questm_1done = 1;
        }
        else if (icount <= 0)
        {
```

```
        questmissText.text = "Cari air minum (0/2)";
        questmissText.color = Color.yellow;
        questm_1done = 0;
    }
}
else if (child.gameObject.tag == "hot water")
{
    string c = child.Find("TextICount").GetComponent<Text>().text;
    int icount = System.Int32.Parse(c);
    if (icount >= 2)
    {
        btn.GetComponent<Image>().enabled = true;
        questmissText.text = "Cari air minum (2/2)";
        questmiss1Text.text = "Bawa ke tempat Paman Gru";
        questmiss1Text.color = Color.yellow;
        btn.GetComponent<Image>().sprite = textureop;
    }
    else if (icount <= 1)
    {
        questmissText.text = "Cari air minum (1/2)";
        questmissText.color = Color.yellow;
    }
    else if (icount <= 0)
    {
        questmissText.text = "Cari air minum (0/2)";
        questmissText.color = Color.yellow;
    }
```

```
}
else if (child.gameObject.tag == "red water")
{
   string c = child.Find("TextICount").GetComponent<Text>().text;
   int icount = System.Int32.Parse(c);
   if (icount >= 2)
   {
      btn.GetComponent<Image>().enabled = true;
      questmissText.text = "Cari air minum (2/2)";
      questmiss1Text.text = "Bawa ke tempat Paman Gru";
      questmiss1Text.color = Color.yellow;
      btn.GetComponent<Image>().sprite = textureop;
   }
   else if (icount <= 1)
   {
      questmissText.text = "Cari air minum (1/2)";
      questmissText.color = Color.yellow;
   }
   else if (icount <= 0)
   {
      questmissText.text = "Cari air minum (0/2)";
      questmissText.color = Color.yellow;
   }
}
else if (child.gameObject.tag == "green water")
{
   string c = child.Find("TextICount").GetComponent<Text>().text;
```

```
        int icount = System.Int32.Parse(c);
        if (icount >= 2)
        {
            btn.GetComponent<Image>().enabled = true;
            questmissText.text = "Cari air minum (2/2)";
            questmiss1Text.text = "Bawa ke tempat Kakek Legenda";
            questmiss1Text.color = Color.yellow;
            btn.GetComponent<Image>().sprite = textureop;
        }
        else if (icount <= 1)
        {
            questmissText.text = "Cari air minum (1/2)";
            questmissText.color = Color.yellow;
        }
        else if (icount <= 0)
        {
            questmissText.text = "Cari air minum (0/2)";
            questmissText.color = Color.yellow;
        }
    }
if (m2 == true)
{
    if (questcount == 0)
    {
        questmissText.text = "Cari 2 jenis makanan (0/2)";
        questmissText.color = Color.yellow;
    }
```

```
    if (questcount == 1)
    {
        questmissText.text = "Cari 2 jenis makanan (1/2)";
        questmissText.color = Color.yellow;
    }
    else if (questcount > 1)
    {
        questmissText.text = "Cari 2 jenis makanan (2/2)";
        btn.GetComponent<Image>().enabled = true;
        questmiss1Text.text = "Bawa ke tempat Paman Gru";
        questmiss1Text.color = Color.yellow;
        btn.GetComponent<Image>().sprite = textureop;
    }
}
if (m3 == true)
{
    if (questm_3done == 0)
    {
        questmissText.text = "Cari tumbuhan tidak beracun dengan tes (0/1)";
        questmissText.color = Color.yellow;
    }
    else if (questm_3done >= 1)
    {
        questmissText.text = "Cari tumbuhan tidak beracun dengan tes (1/1)";
        btn.GetComponent<Image>().enabled = true;
        questmiss1Text.text = "Bawa ke tempat Paman Gru";
        questmiss1Text.color = Color.yellow;
```

```
        btn.GetComponent<Image>().sprite = textureop;
    }
    if (questmissText.text == "Cari tumbuhan tidak beracun dengan tes (0/1)")
    {
        btn.GetComponent<Image>().enabled = false;
    }
}

if (m4 == true)
{
    if (questm_4count >= 3)
    {
        questmissText.text = "Cari 3 jenis makanan dengan semua metode (3/3)";
        btn.GetComponent<Image>().enabled = true;
        questmiss1Text.text = "Bawa ke tempat Paman Gru";
        questmiss1Text.color = Color.yellow;
        btn.GetComponent<Image>().sprite = textureop;
    }
    else if (questm_4count >= 2)
    {
        questmissText.text = "Cari 3 jenis makanan dengan semua metode (2/3)";
        btn.GetComponent<Image>().enabled = false;
        questmissText.color = Color.yellow;
    }
    else if (questm_4count >= 1)
    {
        questmissText.text = "Cari 3 jenis makanan dengan semua metode (1/3)";
```

```
            btn.GetComponent<Image>().enabled = false;
            questmissText.color = Color.yellow;
        }
        else if (questm_4count >= 0)
        {
            questmissText.text = "Cari 3 jenis makanan dengan semua metode (0/3)";
            btn.GetComponent<Image>().enabled = false;
            questmissText.color = Color.yellow;
        }
    }


    // memunculkan gambar berdasarkan nama objek yg sedang di trigger
    if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("Pear"))
        {
            FruitImgFrame.texture = null;
            FruitImgFrame.texture = pearImg;
        }
        Else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("nanas
merah"))
        {
            FruitImgFrame.texture = null;
            FruitImgFrame.texture = nanasmerahImg;
        }
        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("good
water"))
```

```
        {
            FruitImgFrame.texture = null;
            FruitImgFrame.texture = goodWater;
        }
        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("hot water"))
        {
            FruitImgFrame.texture = null;
            FruitImgFrame.texture = hotWater;
        }
        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("red water"))
        {
            FruitImgFrame.texture = null;
            FruitImgFrame.texture = redWater;
        }
        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("green
water"))
        {
            FruitImgFrame.texture = null;
            FruitImgFrame.texture = greenWater;
        }
        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("gympie"))
        {
            FruitImgFrame.texture = null;
            FruitImgFrame.texture = gympie;
        }
```

```
        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("gympiedau
n"))

        {

           FruitImgFrame.texture = null;

           FruitImgFrame.texture = gympiedaun;

        }

        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("Kiwi"))

        {

           FruitImgFrame.texture = null;

           FruitImgFrame.texture = kiwi;

        }

        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("nanas
merah1"))

        {

           FruitImgFrame.texture = null;

           FruitImgFrame.texture = nanasmerahImg;

        }

        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("berberys"))

        {

           FruitImgFrame.texture = null;

           FruitImgFrame.texture = berberys;

        }

        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("algae"))

        {

           FruitImgFrame.texture = null;
```

```
            FruitImgFrame.texture = algae;
        }
        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("gympieplan
t"))
        {
            FruitImgFrame.texture = null;
            FruitImgFrame.texture = gympieplant;
        }
        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("hemlock"))
        {
            FruitImgFrame.texture = null;
            FruitImgFrame.texture = hemlock;
        }
        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("snakeroot")
)
        {
            FruitImgFrame.texture = null;
            FruitImgFrame.texture = snakeroot;
        }
    else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("pekawai"))
    {
        FruitImgFrame.texture = null;
        FruitImgFrame.texture = pekawai;
    }
    else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("jamur"))
```

```
        {
            FruitImgFrame.texture = null;
            FruitImgFrame.texture = jamur;
        }
        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("umbi1"))
        {
            FruitImgFrame.texture = null;
            FruitImgFrame.texture = umbi;
        }
        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("lahung"))
        {
            FruitImgFrame.texture = null;
            FruitImgFrame.texture = lahung;
        }
        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("umbi2"))
        {
            FruitImgFrame.texture = null;
            FruitImgFrame.texture = umbi2;
        }
        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("blackberry"
))
        {
            FruitImgFrame.texture = null;
            FruitImgFrame.texture = blackberry;
        }
```

```
        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("whiteberry"
))

        {

            FruitImgFrame.texture = null;

            FruitImgFrame.texture = whiteberry;

        }

        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("yellowberry
"))

        {

            FruitImgFrame.texture = null;

            FruitImgFrame.texture = yellowberry;

        }

        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("daunkapas"
))

        {

            FruitImgFrame.texture = null;

            FruitImgFrame.texture = daunkapas;

        }

        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("toxicsap"))

        {

            FruitImgFrame.texture = null;

            FruitImgFrame.texture = toxicsap;

        }

        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("nanas
merah2"))

        {
```

```
            FruitImgFrame.texture = null;
            FruitImgFrame.texture = nanasmerahImg;
        }
        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("busuk"))
        {
            FruitImgFrame.texture = null;
            FruitImgFrame.texture = pekawai;
        }
    }
// Fungsi untuk mengonsumsi logistik
    public void consume()
    {
        raycastedObj.GetComponent<ItemsProperties>().Interaction();
        if (raycastedObj.GetComponent<ItemsProperties>().food == true)
        {
            raycastedObj.SetActive(false);
            AnimPlayer.SetBool("Eat", true);
            eatStop = true;
        }
        else
        {
            raycastedObj.SetActive(true);
        }
        camera.enabled = true;
        camera1.enabled = true;
        if (raycastedObj.GetComponent<ItemsProperties>().water == true)
```

```
        {
            AnimPlayer.SetBool("Drink", true);
            drinkstop = true;
        }
        if (Cursor.visible == true)
        {
            Cursor.visible = false;
            Cursor.lockState = CursorLockMode.Locked;
        }
        animdecbox.SetBool("isOpen", false);
        animtextnotif.SetBool("isOpen", true);
        tf = true;
    }
// Fungsi untuk mengambil logistik dan menaruhnya ke inventori
    public void pickup()
    {
        playerChar.SetActive(true);
        crossHairObj.SetActive(false);
        animdecbox.SetBool("isOpen", false);
        animnotifpick.SetBool("isOpen", true);
        tf = true;
        camera.enabled = true;
        camera1.enabled = true;
        AnimPlayer.SetBool("Gather", true);
        gatherstop = true;
        if (Cursor.visible == true)
        {
```

```
            Cursor.visible = false;
            Cursor.lockState = CursorLockMode.Locked;
        }
        if (raycastedObj.GetComponent<ItemsProperties>().food == true)
        {
            raycastedObj.SetActive(false);
        }
        // mengecek icon yg sudah ada di children Objek
        foreach (Transform child in invenPanel.transform)
        {
            // ketika item sudah ada di inven
            if(child.gameObject.tag ==
raycastedObj.GetComponent<ItemsProperties>().itemName)
            {
                string c = child.Find("TextICount").GetComponent<Text>().text;
                int icount = System.Int32.Parse(c) + 1;
                child.Find("TextICount").GetComponent<Text>().text = "" + icount;
                return
            }
        }
        // Belum ada di inven
        GameObject i;
        if (raycastedObj.GetComponent<ItemsProperties>().itemName == "red
apple")
        {
            i = Instantiate(invenIcons[0]);
            i.transform.SetParent(invenPanel.transform);
        }
```

```
        else if (raycastedObj.GetComponent<ItemsProperties>().itemName ==
"green apple")
        {
            i = Instantiate(invenIcons[1]);
            i.transform.SetParent(invenPanel.transform);
        }
        else if (raycastedObj.GetComponent<ItemsProperties>().itemName ==
"yellow apple")
        {
            i = Instantiate(invenIcons[2]);
            i.transform.SetParent(invenPanel.transform);
        }
    }
// Fungsi pengetesan metode iritasi
    public void TesIritasi()
    {
        animdecbox1.SetBool("isOpen", false);
        animdecbox1_5.SetBool("isOpen", false);
        animdecbox2.SetBool("isOpen", true);
    }
// Fungsi pengetesan menggunakan warna getah
    public void TesWarnaGetah()
    {
        animdecbox1_5.SetBool("isOpen", false);
        AnimPlayer.SetBool("Tes", true);
        tes1 = true;
        if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("nanas
merah2"))
```

```
        {
            ntfTes.text = null;
            ntfTes.text = "pure";
        }
        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("pekawai"))
        {
            ntfTes.text = null;
            ntfTes.text = "pure";
        }
        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("jamur"))
        {
            ntfTes.text = null;
            ntfTes.text = "pure";
        }
        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("umbi1"))
        {
            ntfTes.text = null;
            ntfTes.text = "milky";
        }
        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("lahung"))
        {
            ntfTes.text = null;
            ntfTes.text = "pure";
        }
```

```
        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("umbi2"))

        {

            ntfTes.text = null;

            ntfTes.text = "milky";

        }

        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("blackberry"
))

        {

            ntfTes.text = null;

            ntfTes.text = "pure";

        }

        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("whiteberry"
))

        {

            ntfTes.text = null;

            ntfTes.text = "discovered";

        }

        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("yellowberry
"))

        {

            ntfTes.text = null;

            ntfTes.text = "discovered";

        }

        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("daunkapas"
))

        {
```

```
            ntfTes.text = null;
            ntfTes.text = "pure";
        }
        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("toxicsap"))
        {
            ntfTes.text = null;
            ntfTes.text = "discovered";
        }
        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("busuk"))
        {
            ntfTes.text = null;
            ntfTes.text = "pure";
        }
    }
// Beberapa contoh fungsi pengetesan menggunakan rasa
    public void TesRasa()
    {
        animdecbox1_5.SetBool("isOpen", false);
        AnimPlayer.SetBool("Eat", true);
        tes2 = true;

        if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("nanas
merah2"))
        {
            ntfTes.text = null;
            ntfTes.text = "Setelah mencicipi sedikit, terasa manis";
```

```
        }
        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("pekawai"))
        {
            ntfTes.text = null;
            ntfTes.text = "Setelah mencicipi sedikit, terasa manis";
        }
// Beberapa contoh fungsi pengujian logistic menggunakan bau
    public void TesBau()
    {
        animdecbox1_5.SetBool("isOpen", false);
        AnimPlayer.SetBool("Tes", true);
        tes3 = true;

        if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("nanas
merah2"))
        {
            ntfTes.text = null;
            ntfTes.text = "Tidak tercium bau yang aneh";
        }
        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("pekawai"))
        {
            ntfTes.text = null;
            ntfTes.text = "Tidak tercium bau yang aneh";
        }
// Beberapa contoh fungsi pengecekan menggunakan metode iritasi sebelum 8 jam
perut kosong
```

```
    public void CekPlantinSiku()
    {
        animdecbox2.SetBool("isOpen", false);
        AnimPlayer.SetBool("Tes", true);
        cek2 = true;

        if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("nanas
merah1"))
        {
            ntfTes.text = null;
            ntfTes.text = "15 menit telah berlalu, Tidak ada efek";
        }
        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("berberys"))
        {
            ntfTes.text = null;
            ntfTes.text = "15 menit telah berlalu, Tidak ada efek";
        }
        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("algae"))
        {
            ntfTes.text = null;
            ntfTes.text = "15 menit telah berlalu, Tidak ada efek";
        }
    // Beberapa contoh fungsi pengujian logistik menggunakan bibir
    public void CekdiBibir()
    {
        animdecbox2.SetBool("isOpen", false);
```

```
        AnimPlayer.SetBool("Tes", true);
        cek3 = true;

        if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("nanas
merah1"))
        {
            ntfTes.text = null;
            ntfTes.text = "3 menit telah berlalu, Tidak ada efek";
        }
        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("berberys"))
        {
            ntfTes.text = null;
            ntfTes.text = "3 menit telah berlalu, Tidak ada efek";
        }
    // Beberapa contoh pengujian iritasi dari logistic menggunakan Lidah
    public void CekdiLidah()
    {
        animdecbox3.SetBool("isOpen", false);
        AnimPlayer.SetBool("Tes", true);
        cek4 = true;

        if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("nanas
merah1"))
        {
            ntfTes.text = null;
            ntfTes.text = "15 menit telah berlalu, Tidak ada efek";
```

```
        }
        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("berberys"))
        {
            ntfTes.text = null;
            ntfTes.text = "15 menit telah berlalu, Tidak ada efek";
        }
        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("algae"))
        {
            ntfTes.text = null;
            ntfTes.text = "15 menit telah berlalu, Tidak ada efek";
        }
    // Beberapa contoh pengujian iritasi dari logistic dengan mengunyah
    public void CekKunyah()
    {
        animdecbox4.SetBool("isOpen", false);
        AnimPlayer.SetBool("Tes", true);
        cek5 = true;

        if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("nanas
merah1"))
        {
            ntfTes.text = null;
            ntfTes.text = "8 jam telah berlalu, Tidak ada efek";
        }
        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("berberys"))
```

```
        {
            ntfTes.text = null;
            ntfTes.text = "8 jam telah berlalu, Mulut gata - gatal";
        }
    // Fungsi penanganan setelah terjadi iritasi
     public void Muntahkan()
     {
        animdecbox5.SetBool("isOpen", false);
        AnimPlayer.SetBool("Drink", true);
        drinkstop = true;
     }
// Beberapa contoh fungsi pengujian iritasi dengan memakan seperempat cup
     public void makanperempat()
     {
        animdecbox6.SetBool("isOpen", false);
        AnimPlayer.SetBool("Eat", true);
        cek6 = true;

        if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("nanas
merah1"))
        {
            ntfTes.text = null;
            ntfTes.text = "8 jam telah berlalu, Tidak ada efek";
        }
        else if
(raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("berberys"))
        {
```

```
            ntfTes.text = null;
            ntfTes.text = "8 jam telah berlalu, Mulut gata - gatal";
        }
    void crossHairActive()
    {
        crossHair.color = Color.green;
        IndikatorImg.color = Color.green;
    }
    void crossHairNormal()
    {
        crossHair.color = Color.white;
        IndikatorImg.color = Color.white;
    }
}
```

4. Class DialogueDeclare : Memunculkan dialogue jika system memanggil

```
public class DialogueDeclare : MonoBehaviour
{
    public void TriggerDialogue()
    {


        if (storyname.text.Equals("Cari air minum (see details)"))
        {
            qbanim.SetBool("isOpen", true);
        }

    }

    public void closeQuestBox()
    {
        qbanim.SetBool("isOpen", false);
    }
}
```

5. Class DialogueManager : Memunculkan animasi kalimat pada dialog keluar, dan memunculkan kalimat selanjutnya.

```
public class DialogueManager : MonoBehaviour {
   // Berfungsi untuk memunculkan dialog ketika game baru dimulai
   void Start()
   {
      sentences = new Queue<string>();
      StartDialogue(diialogue);
      btn.GetComponent<Image>().sprite = textureop;
   }
   public void StartDialogue (Dialogues dialogue)
   {
      camera.enabled = false;
      camera1.enabled = false;
      ui_animator.SetBool("isOpen", false);
      indikator_anim.SetBool("isOpen", false);
      if (Cursor.visible == false)
         {
            Cursor.visible = true;
            Cursor.lockState = CursorLockMode.Confined;
         }
      animator.SetBool("isOpen", true);
      nameText.text = dialogue.name;
      sentences.Clear();
      pictframe.texture = null;
      pictframe.texture = dialogue.npcimage;
      foreach(string sentence in dialogue.sentences)
      {
         sentences.Enqueue(sentence);
      }
      DisplayNextSentence();
   }

   public void DisplayNextSentence()
   {
      if (sentences.Count == 0)
      {
         EndDialogue();
         return;
      }
      string sentence = sentences.Dequeue();
      StopAllCoroutines();
      StartCoroutine(TypeSentence(sentence));
   }
   IEnumerator TypeSentence(string sentence)
   {
      dialogueText.text = "";
      foreach(char letter in sentence.ToCharArray())
```

```
    {
        dialogueText.text += letter;
        yield return null;
    }
}

public void EndDialogue()
{
    animator.SetBool("isOpen", false);
    ui_animator.SetBool("isOpen", true);
    indikator_anim.SetBool("isOpen", true);
    camera.enabled = true;
    camera1.enabled = true;
    inProgress = true;
    if (Cursor.visible == true)
    {
        Cursor.visible = false;
        Cursor.lockState = CursorLockMode.Locked;
    }
    if(storyname.text.Equals("Mission 1"))
    {
        storyname.text = null;
        storyname.text = "Cari air minum (0/2)";
        storyname.color = Color.yellow;
        btn.GetComponent<Image>().sprite = textureop;
    }
    else if(storyname.text.Equals("Mission 2"))
    {
        storyname.text = null;
        storyname.text = "Cari 2 jenis makanan (0/2)";
        storyname.color = Color.yellow;
        btn.GetComponent<Image>().sprite = textureop;
    }
    else if (storyname.text.Equals("Mission 3"))
    {
        storyname.text = null;
        storyname.text = "Cari makanan tidak beracun dengan tes (0/1)";
        storyname.color = Color.yellow;
        btn.GetComponent<Image>().sprite = textureop;
    }
    else if (storyname.text.Equals("Mission 4"))
    {
        storyname.text = null;
        storyname.text = "Cari 3 jenis makanan dengan semua metode (0/3)";
        storyname.color = Color.yellow;
        btn.GetComponent<Image>().sprite = textureop;
```

```
        }
    }
}
```

6. Class Dialogue : Memberi variabel yang ada didalam dialog

```
public class Dialogues {

    public string name;
    public Texture2D npcimage;
    [TextArea(3, 10)]
    public string[] sentences;
}
```

7. Class EatFromInven : Mengurangi jumlah item yang ada di invent ketika diklik

```
public class EatFromInven : MonoBehaviour {
    // berfungsi untuk membuang makanan dari inven
    void Start()
    {
        playerVital = GetComponent<PlayerVitality>();
        daycontrol = GetComponent<DaysControll>();
    }
    public void consumeFInven()
    {
        if
(System.Int32.Parse(this.transform.Find("TextICount").GetComponent<Text>().t
ext) >= 1)
        {
            int tcount =
System.Int32.Parse(this.transform.Find("TextICount").GetComponent<Text>().te
xt) - 1;
            this.transform.Find("TextICount").GetComponent<Text>().text = "" +
tcount;
            cc.inven = tcount;
            if (food)
            {
                playerVital.energySlider.value += valueE;
                playerVital.cairanSlider.value += valueC;
                playerVital.kesehatanSlider.value -= risk;
            }
            if (water)
            {
                playerVital.cairanSlider.value += valueC;
                playerVital.kesehatanSlider.value -= risk;
            }
            if (sleep)
            {
```

```
            //playerVital.kebugaranSlider.value += value;
            daycontrol.currentTimesOfDay += 0.25f;
            daycontrol.sun.intensity += 0.25f;
        }
    }
    else if
(System.Int32.Parse(this.transform.Find("TextICount").GetComponent<Text>().t
ext) <= 0)
    {

        Destroy(this.gameObject);
    }
  }
}
```

8. Class ItemsProperties : Memberikan feedback terhadap interaksi dari pemain melalui perubahan status bar.

```
public class ItemsProperties : MonoBehaviour {
  // Memberi keterangan pada objek logistik
  public void Interaction()
  {
    if (food)
    {
      playerVital.energySlider.value += valueE;
      playerVital.cairanSlider.value += valueC;
      playerVital.kesehatanSlider.value -= risk;
    }
    if (water)
    {
      playerVital.cairanSlider.value += valueC;
      playerVital.kesehatanSlider.value -= risk;
    }
    if (sleep)
    {
      //playerVital.kebugaranSlider.value += value;
      daycontrol.currentTimesOfDay += 0.25f;
      daycontrol.sun.intensity += 0.25f;
    }

  }
}
```

9. Class PlayerVitality : Mengatur vitalitas pemain secara real time

```
public class PlayerVitality : MonoBehaviour {
```

```
    // Berfungsi membuat bar cairan dan energi berkurang secara real time, dan energi berkurang saat lari beserta kesehatan berkurang saat cairan atau energi habis

    // Use this for initialization

    void Start () {

        // max value controller

        staminaFallRate = 1;

        staminaRegenRate = 1;

        charConroller = GetComponent<Rigidbody>();

        playerController = GetComponent<ThirdPersonCharacter>();

            }

            // Update is called once per frame

            void Update () {

        if(playrDeath == true)

        {

            dyingTime -= Time.deltaTime / 1;

            if(dyingTime <= 0)

            {

                SceneManager.LoadScene(3);

            }

        }

        if (ondelay == true)

        {

            delay -= Time.deltaTime / 1;

            if (delay <= 0)

            {

                animChar.SetBool("Death", true);

                playrDeath = true;
```

```
            ondelay = false;
        }
    }
    if (staminaSlider.value <= 0 && energySlider.value <=0 &&
cairanSlider.value <= 0)
    {
        kesehatanSlider.value -= Time.deltaTime / kesehatanFallRate * 3;
    }
    else if (staminaSlider.value <= 0 && energySlider.value <= 0)
    {
        kesehatanSlider.value -= Time.deltaTime / kesehatanFallRate * 2;
    }
    else if (staminaSlider.value <= 0 && cairanSlider.value <= 0)
    {
        kesehatanSlider.value -= Time.deltaTime / kesehatanFallRate * 2;
    }
    else if (energySlider.value <= 0 && cairanSlider.value <= 0)
    {
        kesehatanSlider.value -= Time.deltaTime / kesehatanFallRate * 2;
    }
    else if (staminaSlider.value <= 0 || energySlider.value <= 0 ||
cairanSlider.value <= 0)
    {
        kesehatanSlider.value -= Time.deltaTime / kesehatanFallRate;
    }
    // death controller
    if (kesehatanSlider.value <= 0)
    {
```

```
    playerDeath();
}
if (energySlider.value >= 0)
{
    energySlider.value -= Time.deltaTime / energyFallRate;
}
else if (energySlider.value >= maxEnergy)
{
    energySlider.value = maxEnergy;
}
else if (energySlider.value <= 0)
{
    energySlider.value = 0;
}
// cairan controller
if (cairanSlider.value >= 0)
{
    cairanSlider.value -= Time.deltaTime / cairanFallRate;
}
else if (cairanSlider.value >= maxCairan)
{
    cairanSlider.value = maxCairan;
}
else if (cairanSlider.value <= 0)
{
    cairanSlider.value = 0;
}
```

```
// stamina controller

if (!Input.GetKey(KeyCode.LeftShift) && charConroller.velocity.magnitude
> 0 && Input.GetKey(KeyCode.UpArrow) || !Input.GetKey(KeyCode.LeftShift)
&& charConroller.velocity.magnitude > 0 && Input.GetKey(KeyCode.W))

{

   staminaSlider.value -= Time.deltaTime / staminaFallRate *
staminaFallMult;

}

else if(charConroller.velocity.magnitude > 0 &&
Input.GetKey(KeyCode.LeftShift))

{

   staminaSlider.value += Time.deltaTime / (staminaRegenRate *
staminaRegenMult / 2);

}

else

{

   staminaSlider.value += Time.deltaTime / staminaRegenRate *
staminaRegenMult;

}

if (staminaSlider.value >= maxStamina)

{

   staminaSlider.value = maxStamina;

}

else if(staminaSlider.value <= 0)

{

   staminaSlider.value = 0;

   playerController.m_MovingTurnSpeed =
playerController.m_StationaryTurnSpeed;

   staminaSlider.value += Time.deltaTime / (staminaRegenRate *
staminaRegenMult / 2);
```

```
        }

        else if(staminaSlider.value >= 0)

        {

            playerController.m_MovingTurnSpeed =
playerController.m_MovingTurnSpeedNorm;

        }

    }

}
```

10. Class WallDetection : Membatasi pemain dalam menjelajahi terrain

```
public class WallDetection : MonoBehaviour {
    public ConsumptionControll cc;
    void OnCollisionEnter()
    {
        if
(cc.raycastedObj.GetComponent<ItemsProperties>().itemName.Equals("wall"))
        {
            cc.notifWall.SetActive(true);
        }
        else
        {
            cc.notifWall.SetActive(false);
        }


    }
}
```

11. Class ThirdPersonCharacters : Mengontrol gerakan, perpindahan *direction* dan animasi karakter.

```
public class ThirdPersonCharacter : MonoBehaviour
            {
        // Berfungsi untuk menggerakan posisi karakter dan memanggil berbagai
animasi tergantung kondisi dan perintah

        void Start()
                    {
                            m_Animator = GetComponent<Animator>();
                            m_Rigidbody = GetComponent<Rigidbody>();
                            m_Capsule = GetComponent<CapsuleCollider>();
                            m_CapsuleHeight = m_Capsule.height;
```

```
                m_CapsuleCenter = m_Capsule.center;

                m_Rigidbody.constraints =
RigidbodyConstraints.FreezeRotationX | RigidbodyConstraints.FreezeRotationY |
RigidbodyConstraints.FreezeRotationZ;
                m_OrigGroundCheckDistance = m_GroundCheckDistance;
        }


        public void Move(Vector3 move, bool crouch, bool jump)
        {

                // convert the world relative moveInput vector into a local-
relative
                // turn amount and forward amount required to head in the
desired
                // direction.
                if (move.magnitude > 1f) move.Normalize();
                move = transform.InverseTransformDirection(move);
                CheckGroundStatus();
                move = Vector3.ProjectOnPlane(move,
m_GroundNormal);
                m_TurnAmount = Mathf.Atan2(move.x, move.z);
                m_ForwardAmount = move.z;

                ApplyExtraTurnRotation();

                // control and velocity handling is different when grounded
and airborne:
                if (m_IsGrounded)
                {
                        HandleGroundedMovement(crouch, jump);
                }
                else
                {
                        HandleAirborneMovement();
                }

                ScaleCapsuleForCrouching(crouch);
                PreventStandingInLowHeadroom();

                // send input and other state parameters to the animator
                UpdateAnimator(move);
        }
```

```
            void ScaleCapsuleForCrouching(bool crouch)
            {
                if (m_IsGrounded && crouch)
                {
                    if (m_Crouching) return;
                    m_Capsule.height = m_Capsule.height / 2f;
                    m_Capsule.center = m_Capsule.center / 2f;
                    m_Crouching = true;
                }
                else
                {
                    Ray crouchRay = new Ray(m_Rigidbody.position +
Vector3.up * m_Capsule.radius * k_Half, Vector3.up);
                    float crouchRayLength = m_CapsuleHeight -
m_Capsule.radius * k_Half;
                    if (Physics.SphereCast(crouchRay,
m_Capsule.radius * k_Half, crouchRayLength, Physics.AllLayers,
QueryTriggerInteraction.Ignore))
                    {
                        m_Crouching = true;
                        return;
                    }
                    m_Capsule.height = m_CapsuleHeight;
                    m_Capsule.center = m_CapsuleCenter;
                    m_Crouching = false;
                }
            }

            void PreventStandingInLowHeadroom()
            {
                // prevent standing up in crouch-only zones
                if (!m_Crouching)
                {
                    Ray crouchRay = new Ray(m_Rigidbody.position +
Vector3.up * m_Capsule.radius * k_Half, Vector3.up);
                    float crouchRayLength = m_CapsuleHeight -
m_Capsule.radius * k_Half;
                    if (Physics.SphereCast(crouchRay,
m_Capsule.radius * k_Half, crouchRayLength, Physics.AllLayers,
QueryTriggerInteraction.Ignore))
                    {
                        m_Crouching = true;
                    }
                }
            }
            void UpdateAnimator(Vector3 move)
```

```
                {
                        // update the animator parameters
                        m_Animator.SetFloat("Forward", m_ForwardAmount, 0.1f,
Time.deltaTime);
                        m_Animator.SetFloat("Turn", m_TurnAmount, 0.1f,
Time.deltaTime);
                        m_Animator.SetBool("Crouch", m_Crouching);
                        m_Animator.SetBool("OnGround", m_IsGrounded);
                        if (!m_IsGrounded)
                        {
                                m_Animator.SetFloat("Jump",
m_Rigidbody.velocity.y);
                        }

                        // calculate which leg is behind, so as to leave that leg
trailing in the jump animation
                        // (This code is reliant on the specific run cycle offset in our
animations,
                        // and assumes one leg passes the other at the normalized
clip times of 0.0 and 0.5)
                        float runCycle =
                                Mathf.Repeat(

        m_Animator.GetCurrentAnimatorStateInfo(0).normalizedTime +
m_RunCycleLegOffset, 1);
                        float jumpLeg = (runCycle < k_Half ? 1 : -1) *
m_ForwardAmount;
                        if (m_IsGrounded)
                        {
                                m_Animator.SetFloat("JumpLeg", jumpLeg);
                        }

                        // the anim speed multiplier allows the overall speed of
walking/running to be tweaked in the inspector,
                        // which affects the movement speed because of the root
motion.
                        if (m_IsGrounded && move.magnitude > 0)
                        {
                                m_Animator.speed = m_AnimSpeedMultiplier;
            PlayFootStepAudio();
                        }
                        else
                        {
                                // don't use that while airborne
                                m_Animator.speed = 1;
                        }
```

```
		}


		void HandleAirborneMovement()
		{
			// apply extra gravity from multiplier:
			Vector3 extraGravityForce = (Physics.gravity * 
m_GravityMultiplier) - Physics.gravity;
			m_Rigidbody.AddForce(extraGravityForce);

			m_GroundCheckDistance = m_Rigidbody.velocity.y < 0 ? 
m_OrigGroundCheckDistance : 0.01f;
		}
		void HandleGroundedMovement(bool crouch, bool jump)
		{
			// check whether conditions are right to allow a jump:
			if (jump && !crouch && 
m_Animator.GetCurrentAnimatorStateInfo(0).IsName("Grounded"))
			{
				// jump!
				m_Rigidbody.velocity = new 
Vector3(m_Rigidbody.velocity.x, m_JumpPower, m_Rigidbody.velocity.z);
				m_IsGrounded = false;
				m_Animator.applyRootMotion = false;
				m_GroundCheckDistance = 0.1f;
	PlayJumpSound();
			}
		}

		void ApplyExtraTurnRotation()
		{
			// help the character turn faster (this is in addition to root 
rotation in the animation)
			float turnSpeed = Mathf.Lerp(m_MovingTurnSpeed, 
m_StationaryTurnSpeed, m_ForwardAmount);
			transform.Rotate(0, m_TurnAmount * turnSpeed * 
Time.deltaTime, 0);
		}
		public void OnAnimatorMove()
		{
			// we implement this function to override the default root 
motion.
			// this allows us to modify the positional speed before it's 
applied.
			if (m_IsGrounded && Time.deltaTime > 0)
			{
```

```
                                Vector3 v = (m_Animator.deltaPosition *
m_MoveSpeedMultiplier) / Time.deltaTime;

                                // we preserve the existing y part of the current
velocity.
                                v.y = m_Rigidbody.velocity.y;
                                m_Rigidbody.velocity = v;
            // PlayFootStepAudio();
                        }
                }
                void CheckGroundStatus()
                {
                        RaycastHit hitInfo;
#if UNITY_EDITOR
                        // helper to visualise the ground check ray in the scene view
                        Debug.DrawLine(transform.position + (Vector3.up * 0.1f),
transform.position + (Vector3.up * 0.1f) + (Vector3.down *
m_GroundCheckDistance));
#endif
                        // 0.1f is a small offset to start the ray from inside the
character
                        // it is also good to note that the transform position in the
sample assets is at the base of the character
                        if (Physics.Raycast(transform.position + (Vector3.up *
0.1f), Vector3.down, out hitInfo, m_GroundCheckDistance))
                        {
                                m_GroundNormal = hitInfo.normal;
                                m_IsGrounded = true;
                                m_Animator.applyRootMotion = true;
                        }
                        else
                        {
                                m_IsGrounded = false;
                                m_GroundNormal = Vector3.up;
                                m_Animator.applyRootMotion = false;
                        }
                }
        private void PlayJumpSound()
        {
           m_AudioSource.clip = m_JumpSound;
           m_AudioSource.Play();
        }

        private void PlayFootStepAudio()
        {
           if (!m_IsGrounded)
```

```
            {
                return;
            }
            // pick & play a random footstep sound from the array,
            // excluding sound at index 0
            int n = Random.Range(1, m_FootstepSounds.Length);
            m_AudioSource.clip = m_FootstepSounds[n];
            m_AudioSource.PlayOneShot(m_AudioSource.clip);
            // move picked sound to index 0 so it's not picked next time
            m_FootstepSounds[n] = m_FootstepSounds[0];
            m_FootstepSounds[0] = m_AudioSource.clip;
        }
    }
}
```

Lampiran D. Soal Pretest dan Posttest

Pretest

Nama :

Umur :

1. Warna buah apa yang sebaiknya dihindari untuk dimakan saat tersesat di hutan?

   a. Kuning
   c. Hitam
   b. Hijau
   d. Ungu

2. Bagaimana cara mencari makanan yang tidak beracun di hutan di hutan menggunakan petunjuk hewan?
   a. Makan tumbuhan yang di dekatnya ada hewan mamalia.
   b. Makan tumbuhan yang dilewati hewan mamalia.
   c. Makan tumbuhan yang dimakan hewan mamalia.
   d. Makan tumbuhan yang dikerumuni hewan mamalia.
3. Karakter tumbuhan seperti apa yang sebaiknya tidak dikonsumsi ketika tersesat di hutan?
   a. Mengandung banyak air
   c. Mengandung banyak getah
   b. Mempunyai banyak biji
   d. Tidak mengandung air
4. Ketika tersesat di hutan apa yang sebaiknya dilakukan?
   a. Makan 1 jenis tumbuhan
   c. Hanya minum
   b. Makan berbagai jenis tumbuhan
   d. Makan sedikit jenis tumbuhan
5. Manakah air yang sebaiknya dikonsumsi saat tersesat di hutan berdasarkan baunya?
   a. Berbau wangi
   c. Berbau amis
   b. Tidak berbau
   d. Berbau belerang
6. Dari pernyataan di bawah ini manakah yang sebaiknya tidak dikonsumsi tanpa pengolahaan yang benar saat tersesat di hutan?

1. Jamur
2. Umbi – umbian
3. Tumbuhan yang masih muda
4. Arbei

Jawab :

a. 1 dan 4
b. 1 dan 2
c. 1 dan 3
d. 1,2,3 dan 4

7. Manakah air yang baik untuk dikonsumsi saat tersesat di hutan berdasarkan warna?
   a. Berwarna merah
   b. Berwarna coklat
   c. Berwarna hijau
   d. Tidak berwarna
8. Manakah tumbuhan yang baik untuk dikonsumsi saat tersesat di hutan berdasarkan bau?
   a. Berbau tidak sedap
   b. Berbau menyengat
   c. Tidak berbau
   d. Berbau almond
9. Berdasarkan suhunya manakah air yang sebaiknya tidak dikonsumsi saat tersesat dihutan?
   a. Air dingin
   b. Air hangat
   c. Air panas
   d. Air bersuhu sedang
10. Apa langkah paling baik saat mencari makan di hutan ketika tersesat?
    a. Makan tumbuhan yang ditemui
    b. Makan tumbuhan yang terlihat mencolok
    c. Makan tumbuhan yang sudah dikenal
    d. Asal memakan tumbuhan
11. Mamalia apa yang paling cocok untuk dijadikan acuan dalam mencari makanan yang tidak beracun saat tersesat di hutan?
    a. Beruang
    Kijang
    c. Kera
    d. Banteng
12. Tumbuhan apa yang perlu dihindari untuk dikonsumsi saat tersesat di hutan berdasarkan rasa?
    a. Tumbuhan berasa pahit
    b. Tumbuhan berasa manis
    c. Tumbuhan berasa masam
    d. Tumbuhan berasa tawar
13. Dari pernyataan di bawah ini mana berry yang dapat dikonsumsi?

1. Buah berry berwarna putih.
2. Buah berry berwarna kuning.
3. Buah berry berwarna hitam.

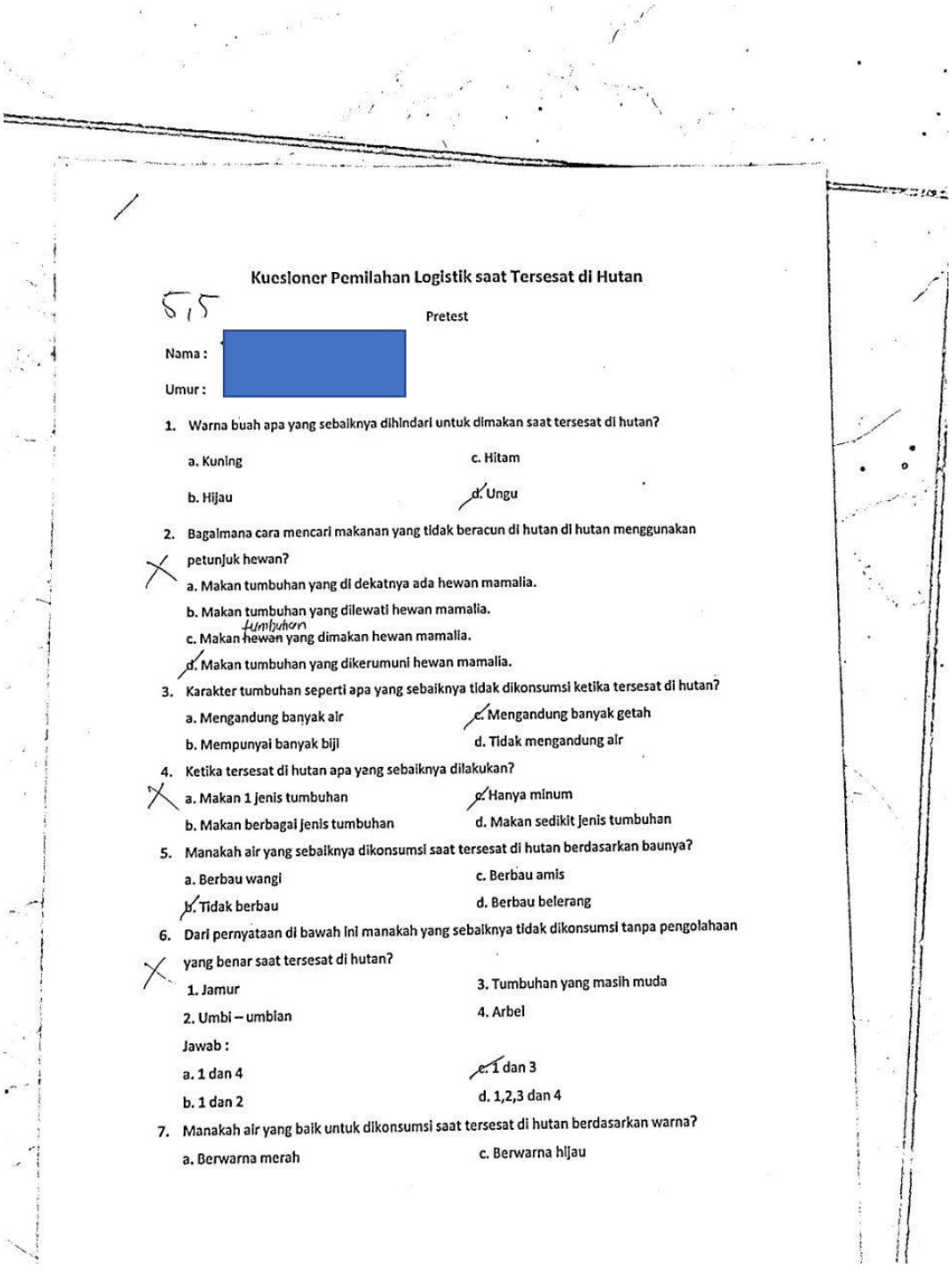
Jawab :

a. 1 dan 2 c. 3
b. 2 d. 1

14. Berdasarkan bentuk daunnya tumbuhan seperti apa yang sebaiknya dihindari data tersesat di hutan?
    a. Tumbuhan dengan daun bercabang tiga
    b. Tumbuhan dengan daun sejajar
    c. Tumbuhan dengan daun melengkung
    d. Tumbuhan dengan daun menyirip
15. Untuk melakukan pengetesan iritasi pada tumbuhan, berapa waktu yang dibutuhkan untuk perut dalam keadaan kosong?
    a. 6 jam c. 8 jam
    b. 7 jam d. 9 jam
16. Berdasarkan karakteristik daunnya, manakah tumbuhan yang sebaiknya dihindari?
    a. Tumbuhan dengan daun basah c. Tumbuhan dengan daun berbulu / berduri
    b. Tumbuhan dengan daun bertekstur halus d. Tumbuhan dengan daun bertekstur kasar
17. Sebelum waktu perut dalam keadaan kosong tercapai sesuai prosedur, metode tes iritasi seperti apa yang bisa dilakukan?
    a. Meletakkan bagian tumbuhan di telapak tangan
    b. Meletakkan bagian tumbuhan di bibir
    c. Meletakkan bagian tumbuhan di siku
    d. Meletakkan bagian tumbuhan di lidah
18. Setelah perut sudah kosong selama waktu yang ditentukan, metode tes seperti apa yang bisa dilakukan?

a. Menaruh bagian tumbuhan ke bibir
c. Menaruh bagian tumbuhan ke siku
b. Memakan tumbuahan
d. Menaruh bagian tumbuhan ke lidah

19. Urutan metode pengetesan tumbuhan untuk mengetahui tumbuhan menyebabkan iritasi yang benar adalah?
    a. Taruh bagian tumbuhan ke bibir bagian luar - Taruh bagian tumbuhan ke lidah – Kunyah dan diamkan – Makan 0.25 cup
    b. Taruh bagian tumbuhan ke lidah – Kunyah dan diamkan - Makan 0.25 cup - Taruh bagian tumbuhan ke bibir bagian luar
    c. Makan 0.25 cup - Taruh bagian tumbuhan ke bibir bagian luar - Taruh bagian tumbuhan ke lidah – Kunyah dan diamkan
    d. Kunyah dan diamkan - Makan 0.25 cup - Taruh bagian tumbuhan ke bibir bagian luar - Taruh bagian tumbuhan ke lidah
20. Berdasarkan getahnya tumbuhan seperti apa yang baik untuk dimakan?
    a. Tumbuhan dengan getah yang berganti warna
    b. Tumbuhan dengan getah yang kental
    c. Tumbuhan dengan getah yang sedikit
    d. Tumbuhan dengan getah yang banyak



Kuesioner Pemilahan Logistik saat Tersesat di Hutan

Pretest

Nama :

Umur :

1. Warna buah apa yang sebaiknya dihindari untuk dimakan saat tersesat di hutan?
   a. Kuning
   c. Hitam
   b. Hijau
   d. Ungu
2. Bagaimana cara mencari makanan yang tidak beracun di hutan di hutan menggunakan petunjuk hewan?
   a. Makan tumbuhan yang di dekatnya ada hewan mamalia.
   b. Makan tumbuhan yang dilewati hewan mamalia.
   c. Makan hewan yang dimakan hewan mamalia.
   d. Makan tumbuhan yang dikerumuni hewan mamalia.
3. Karakter tumbuhan seperti apa yang sebaiknya tidak dikonsumsi ketika tersesat di hutan?
   a. Mengandung banyak air
   c. Mengandung banyak getah
   b. Mempunyai banyak biji
   d. Tidak mengandung air
4. Ketika tersesat di hutan apa yang sebaiknya dilakukan?
   a. Makan 1 jenis tumbuhan
   c. Hanya minum
   b. Makan berbagai jenis tumbuhan
   d. Makan sedikit jenis tumbuhan
5. Manakah air yang sebaiknya dikonsumsi saat tersesat di hutan berdasarkan baunya?
   a. Berbau wangi
   c. Berbau amis
   b. Tidak berbau
   d. Berbau belerang
6. Dari pernyataan di bawah ini manakah yang sebaiknya tidak dikonsumsi tanpa pengolahaan yang benar saat tersesat di hutan?
   1. Jamur
   3. Tumbuhan yang masih muda
   2. Umbi – umbian
   4. Arbel

   Jawab :
   a. 1 dan 4
   c. 1 dan 3
   b. 1 dan 2
   d. 1,2,3 dan 4
7. Manakah air yang baik untuk dikonsumsi saat tersesat di hutan berdasarkan warna?
   a. Berwarna merah
   c. Berwarna hijau

b. Berwarna coklat
d. Tidak berwarna

8. Manakah tumbuhan yang baik untuk dikonsumsi saat tersesat di hutan berdasarkan bau?
   a. Berbau tidak sedap
   c. Tidak berbau
   b. Berbau menyengat
   d. Berbau almond
9. Berdasarkan suhunya manakah air yang sebaiknya tidak dikonsumsi saat tersesat dihutan?
   a. Air dingin
   c. Air panas
   b. Air hangat
   d. Air bersuhu sedang
10. Apa langkah paling baik saat mencari makan di hutan ketika tersesat?
    a. Makan tumbuhan yang ditemui
    c. Makan tumbuhan yang sudah dikenal
    b. Makan tumbuhan yang terlihat mencolok
    d. Asal memakan tumbuhan
11. Mamalia apa yang paling cocok untuk dijadikan acuan dalam mencari makanan yang tidak beracun saat tersesat di hutan?
    a. Beruang
    c. Kera
    Kijang
    d. Banteng
12. Tumbuhan apa yang perlu dihindari untuk dikonsumsi saat tersesat di hutan berdasarkan rasa?
    a. Tumbuhan berasa pahit
    c. Tumbuhan berasa masam
    b. Tumbuhan berasa manis
    d. Tumbuhan berasa tawar
13. Dari pernyataan di bawah ini mana berry yang dapat dikonsumsi?
    1. Buah berry berwarna putih.
    2. Buah berry berwarna kuning.
    3. Buah berry berwarna hitam.

    Jawab :
    a. 1 dan 2
    c. 3
    b. 2
    d. 1
14. Berdasarkan bentuk daunnya tumbuhan seperti apa yang sebaiknya dihindari data tersesat di hutan?
    a. Tumbuhan dengan daun bercabang tiga
    b. Tumbuhan dengan daun sejajar
    c. Tumbuhan dengan daun melengkung
    d. Tumbuhan dengan daun menyirip
15. Untuk melakukan pengetesan iritasi pada tumbuhan, berapa waktu yang dibutuhkan untuk perut dalam keadaan kosong?
    a. 6 jam
    c. 8 jam

b. 7 Jam
d. 9 Jam

16. Berdasarkan karakteristik daunnya, manakah tumbuhan yang sebaiknya dihindari?
a. Tumbuhan dengan daun basah
c. Tumbuhan dengan daun berbulu / berduri
b. Tumbuhan dengan daun bertekstur halus
d. Tumbuhan dengan daun bertekstur kasar

17. Sebelum waktu perut dalam keadaan kosong tercapai sesuai prosedur, metode tes iritasi seperti apa yang bisa dilakukan?
a. Meletakkan bagian tumbuhan di telapak tangan
b. Meletakkan bagian tumbuhan di bibir
c. Meletakkan bagian tumbuhan di siku
d. Meletakkan bagian tumbuhan di lidah

18. Setelah perut sudah kosong selama waktu yang ditentukan, metode tes seperti apa yang bisa dilakukan?
a. Menaruh bagian tumbuhan ke bibir
c. Menaruh bagian tumbuhan ke siku
b. Memakan tumbuahan
d. Menaruh bagian tumbuhan ke lidah

19. Urutan metode pengetesan tumbuhan untuk mengetahui tumbuhan menyebabkan iritasi yang benar adalah?
a. Taruh bagian tumbuhan ke bibir bagian luar - Taruh bagian tumbuhan ke lidah – Kunyah dan diamkan – Makan 0.25 cup
b. Taruh bagian tumbuhan ke lidah – Kunyah dan diamkan - Makan 0.25 cup - Taruh bagian tumbuhan ke bibir bagian luar
c. Makan 0.25 cup - Taruh bagian tumbuhan ke bibir bagian luar - Taruh bagian tumbuhan ke lidah – Kunyah dan diamkan
d. Kunyah dan diamkan - Makan 0.25 cup - Taruh bagian tumbuhan ke bibir bagian luar - Taruh bagian tumbuhan ke lidah

20. Berdasarkan getahnya tumbuhan seperti apa yang baik untuk dimakan?
a. Tumbuhan dengan getah yang berganti warna
b. Tumbuhan dengan getah yang kental
c. Tumbuhan dengan getah yang sedikit
d. Tumbuhan dengan getah yang banyak

Posttest

Nama :

Umur :

1. Berdasarkan karakteristik daunnya, manakah tumbuhan yang sebaiknya dihindari?
   a. Tumbuhan dengan daun basah
   b. Tumbuhan dengan daun bertekstur halus
   c. Tumbuhan dengan daun berbulu / berduri
   d. Tumbuhan dengan daun bertekstur kasar
2. Berdasarkan bentuk daunnya tumbuhan seperti apa yang sebaiknya dihindari data tersesat di hutan?
   a. Tumbuhan dengan daun bercabang tiga
   b. Tumbuhan dengan daun sejajar
   c. Tumbuhan dengan daun melengkung
   d. Tumbuhan dengan daun menyirip
3. Karakter tumbuhan seperti apa yang sebaiknya tidak dikonsumsi ketika tersesat di hutan?
   a. Mengandung banyak air
   b. Mempunyai banyak biji
   c. Mengandung banyak getah
   d. Tidak mengandung air
4. Manakah air yang sebaiknya dikonsumsi saat tersesat di hutan berdasarkan baunya?
   a. Berbau wangi
   b. Tidak berbau
   c. Berbau amis
   d. Berbau belerang

5. Manakah air yang baik untuk dikonsumsi saat tersesat di hutan berdasarkan warna?
   a. Berwarna merah
   b. Berwarna coklat
   c. Berwarna hijau
   d. Tidak berwarna
6. Berdasarkan suhunya manakah air yang sebaiknya tidak dikonsumsi saat tersesat dihutan?
   a. Air dingin
   b. Air hangat
   c. Air panas
   d. Air bersuhu sedang
7. Mamalia apa yang paling cocok untuk dijadikan acuan dalam mencari makanan yang tidak beracun saat tersesat di hutan?
   a. Beruang
   Kijang
   c. Kera
   d. Banteng
8. Dari pernyataan di bawah ini mana berry yang dapat dikonsumsi?
   1. Buah berry berwarna putih.
   2. Buah berry berwarna kuning.
   3. Buah berry berwarna hitam.
   Jawab :
   a. 1 dan 2
   b. 2
   c. 3
   d. 1
9. Untuk melakukan pengetesan iritasi pada tumbuhan, berapa waktu yang dibutuhkan untuk perut dalam keadaan kosong?
   a. 6 jam
   b. 7 jam
   c. 8 jam
   d. 9 jam
10. Sebelum waktu perut dalam keadaan kosong tercapai sesuai prosedur, metode tes iritasi seperti apa yang bisa dilakukan?
    a. Meletakkan bagian tumbuhan di telapak tangan
    b. Meletakkan bagian tumbuhan di bibir
    c. Meletakkan bagian tumbuhan di siku
    d. Meletakkan bagian tumbuhan di lidah
11. Urutan metode pengetesan tumbuhan untuk mengetahui tumbuhan menyebabkan iritasi yang benar adalah?

a. Taruh bagian tumbuhan ke bibir bagian luar - Taruh bagian tumbuhan ke lidah – Kunyah dan diamkan – Makan 0.25 cup

b. Taruh bagian tumbuhan ke lidah – Kunyah dan diamkan - Makan 0.25 cup - Taruh bagian tumbuhan ke bibir bagian luar

c. Makan 0.25 cup - Taruh bagian tumbuhan ke bibir bagian luar - Taruh bagian tumbuhan ke lidah – Kunyah dan diamkan

d. Kunyah dan diamkan - Makan 0.25 cup - Taruh bagian tumbuhan ke bibir bagian luar - Taruh bagian tumbuhan ke lidah

12. Berdasarkan getahnya tumbuhan seperti apa yang baik untuk dimakan?
    a. Tumbuhan dengan getah yang berganti warna
    b. Tumbuhan dengan getah yang kental
    c. Tumbuhan dengan getah yang sedikit
    d. Tumbuhan dengan getah yang banyak
13. Setelah perut sudah kosong selama waktu yang ditentukan, metode tes seperti apa yang bisa dilakukan?
    a. Menaruh bagian tumbuhan ke bibir
    c. Menaruh bagian tumbuhan ke siku
    b. Memakan tumbuahan
    d. Menaruh bagian tumbuhan ke lidah
14. Tumbuhan apa yang perlu dihindari untuk dikonsumsi saat tersesat di hutan berdasarkan rasa?
    a. Tumbuhan berasa pahit
    c. Tumbuhan berasa masam
    b. Tumbuhan berasa manis
    d. Tumbuhan berasa tawar
15. Apa langkah paling baik saat mencari makan di hutan ketika tersesat?
    a. Makan tumbuhan yang ditemui
    c. Makan tumbuhan yang sudah dikenal
    b. Makan tumbuhan yang terlihat mencolok
    d. Asal memakan tumbuhan
16. Manakah tumbuhan yang baik untuk dikonsumsi saat tersesat di hutan berdasarkan bau?
    a. Berbau tidak sedap
    c. Tidak berbau
    b. Berbau menyengat
    d. Berbau almond

17. Dari pernyataan di bawah ini manakah yang sebaiknya tidak dikonsumsi tanpa pengolahaan yang benar saat tersesat di hutan?

    1. Jamur
    2. Umbi – umbian
    3. Tumbuhan yang masih muda
    4. Arbei

    Jawab :

    a. 1 dan 4
    b. 1 dan 2
    c. 1 dan 3
    d. 1,2,3 dan 4

18. Ketika tersesat di hutan apa yang sebaiknya dilakukan?

    a. Makan 1 jenis tumbuhan
    b. Makan berbagai jenis tumbuhan
    c. Hanya minum
    d. Makan sedikit jenis tumbuhan

19. Bagaimana cara mencari makanan yang tidak beracun di hutan di hutan menggunakan petunjuk hewan?

    a. Makan tumbuhan yang di dekatnya ada hewan mamalia.
    b. Makan tumbuhan yang dilewati hewan mamalia.
    c. Makan tumbuhan yang dimakan hewan mamalia.
    d. Makan tumbuhan yang dikerumuni hewan mamalia.

20. Warna buah apa yang sebaiknya dihindari untuk dimakan saat tersesat di hutan?

    a. Kuning
    b. Hijau
    c. Hitam
    d. Ungu

7,5

Kuesioner Pemilahan Logistik saat Tersesat di Hutan

Posttest

Nama :

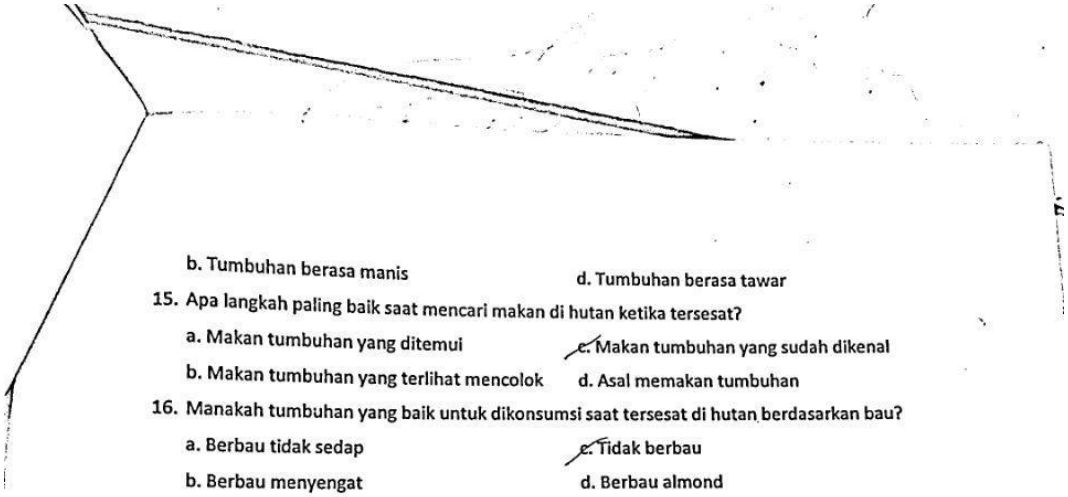
Umur :

1. Berdasarkan karakteristik daunnya, manakah tumbuhan yang sebaiknya dihindari?
   a. Tumbuhan dengan daun basah
   c. Tumbuhan dengan daun berbulu / berduri
   b. Tumbuhan dengan daun bertekstur halus
   d. Tumbuhan dengan daun bertekstur kasar
2. Berdasarkan bentuk daunnya tumbuhan seperti apa yang sebaiknya dihindari data tersesat di hutan?
   a. Tumbuhan dengan daun bercabang tiga
   b. Tumbuhan dengan daun sejajar
   c. Tumbuhan dengan daun melengkung
   d. Tumbuhan dengan daun menyirip
3. Karakter tumbuhan seperti apa yang sebaiknya tidak dikonsumsi ketika tersesat di hutan?
   a. Mengandung banyak air
   c. Mengandung banyak getah
   b. Mempunyai banyak biji
   d. Tidak mengandung air
4. Manakah air yang sebaiknya dikonsumsi saat tersesat di hutan berdasarkan baunya?
   a. Berbau wangi
   c. Berbau amis
   b. Tidak berbau
   d. Berbau belerang
5. Manakah air yang baik untuk dikonsumsi saat tersesat di hutan berdasarkan warna?
   a. Berwarna merah
   c. Berwarna hijau
   b. Berwarna coklat
   d. Tidak berwarna
6. Berdasarkan suhunya manakah air yang sebaiknya tidak dikonsumsi saat tersesat dihutan?
   a. Air dingin
   c. Air panas
   b. Air hangat
   d. Air bersuhu sedang
7. Mamalia apa yang paling cocok untuk dijadikan acuan dalam mencari makanan yang tidak beracun saat tersesat di hutan?
   a. Beruang
   c. Kera
   Kijang
   d. Banteng
8. Dari pernyataan di bawah ini mana berry yang dapat dikonsumsi?
   1. Buah berry berwarna putih.
   2. Buah berry berwarna kuning.
   3. Buah berry berwarna hitam.

   Jawab :

a. 1 dan 2
b. 2
c. 3
d. 1

9. Untuk melakukan pengetesan iritasi pada tumbuhan, berapa waktu yang dibutuhkan untuk perut dalam keadaan kosong?
   a. 6 jam
   b. 7 jam
   c. 8 jam
   d. 9 jam
10. Sebelum waktu perut dalam keadaan kosong tercapai sesuai prosedur, metode tes iritasi seperti apa yang bisa dilakukan?
    a. Meletakkan bagian tumbuhan di telapak tangan
    b. Meletakkan bagian tumbuhan di bibir
    c. Meletakkan bagian tumbuhan di siku
    d. Meletakkan bagian tumbuhan di lidah
11. Urutan metode pengetesan tumbuhan untuk mengetahui tumbuhan menyebabkan iritasi yang benar adalah?
    a. Taruh bagian tumbuhan ke bibir bagian luar - Taruh bagian tumbuhan ke lidah – Kunyah dan diamkan – Makan 0.25 cup
    b. Taruh bagian tumbuhan ke lidah – Kunyah dan diamkan - Makan 0.25 cup - Taruh bagian tumbuhan ke bibir bagian luar
    c. Makan 0.25 cup - Taruh bagian tumbuhan ke bibir bagian luar - Taruh bagian tumbuhan ke lidah – Kunyah dan diamkan
    d. Kunyah dan diamkan - Makan 0.25 cup - Taruh bagian tumbuhan ke bibir bagian luar - Taruh bagian tumbuhan ke lidah
12. Berdasarkan getahnya tumbuhan seperti apa yang baik untuk dimakan?
    a. Tumbuhan dengan getah yang berganti warna
    b. Tumbuhan dengan getah yang kental
    c. Tumbuhan dengan getah yang sedikit
    d. Tumbuhan dengan getah yang banyak
13. Setelah perut sudah kosong selama waktu yang ditentukan, metode tes seperti apa yang bisa dilakukan?
    a. Menaruh bagian tumbuhan ke bibir
    b. Memakan tumbuahan
    c. Menaruh bagian tumbuhan ke siku
    d. Menaruh bagian tumbuhan ke lidah
14. Tumbuhan apa yang perlu dihindari untuk dikonsumsi saat tersesat di hutan berdasarkan rasa?
    a. Tumbuhan berasa pahit
    c. Tumbuhan berasa masam



b. Tumbuhan berasa manis
d. Tumbuhan berasa tawar

15. Apa langkah paling baik saat mencari makan di hutan ketika tersesat?
    a. Makan tumbuhan yang ditemui
    c. Makan tumbuhan yang sudah dikenal
    b. Makan tumbuhan yang terlihat mencolok
    d. Asal memakan tumbuhan
16. Manakah tumbuhan yang baik untuk dikonsumsi saat tersesat di hutan berdasarkan bau?
    a. Berbau tidak sedap
    c. Tidak berbau
    b. Berbau menyengat
    d. Berbau almond
17. Dari pernyataan di bawah ini manakah yang sebaiknya tidak dikonsumsi tanpa pengolahaan yang benar saat tersesat di hutan?
    1. Jamur
    3. Tumbuhan yang masih muda
    2. Umbi – umbian
    4. Arbei
    Jawab :
    a. 1 dan 4
    c. 1 dan 3
    b. 1 dan 2
    d. 1,2,3 dan 4
18. Ketika tersesat di hutan apa yang sebaiknya dilakukan?
    a. Makan 1 jenis tumbuhan
    c. Hanya minum
    b. Makan berbagai jenis tumbuhan
    d. Makan sedikit jenis tumbuhan
19. Bagaimana cara mencari makanan yang tidak beracun di hutan di hutan menggunakan petunjuk hewan?
    a. Makan tumbuhan yang di dekatnya ada hewan mamalia.
    b. Makan tumbuhan yang dilewati hewan mamalia.
    c. Makan hewan (tumbuhan) yang dimakan hewan mamalia.
    d. Makan tumbuhan yang dikerumuni hewan mamalia.
20. Warna buah apa yang sebaiknya dihindari untuk dimakan saat tersesat di hutan?
    a. Kuning
    c. Hitam
    b. Hijau
    d. Ungu

Lampiran E. Dokumentasi