G. M. Refatul Islam
20101482
Section: 9

Task1:

```
#!/usr/bin/env bash

read -r -p "Enter an array: " -a arr
sums=0

if [ ${#arr[*]} -gt 10 ]; then
    echo you have entered more than 10 characters
else
    for i in {0..10}; do
        if [ $((arr[i] % 2)) -eq 0 ] && [
$((arr[i] % 8)) -ne 0 ]; then
            sums=$((arr[i]+sums))
        fi
    done
    echo "Total Sum: ${sums}"
fi

#input should be like this
# 2 3 4 9 0 8 0 ..
```

Task2:

```
read -r -p "Enter a number: " nums

if [ $((nums % 4)) -ne 0 ] && [ $((nums % 5)) -ne
0 ] && [ $((nums % 10)) -eq 0 ]; then
    echo "Rasengan"
fi

if [ $((nums % 5)) -eq 0 ] || [ $((nums % 6)) -eq
0 ]; then
    if [ $((nums % 5)) -eq 0 ] && [ $((nums % 6))
-eq 0 ]; then
        :
    else
        echo "Odama Rasengan"
    fi
fi

if [ $((nums % 5)) -eq 0 ] && [ $((nums % 6)) -eq
0 ]; then
    echo "Rasen Shuriken"
fi
```

Task3:

```


read -r -p "Enter a Number: " nums
```

```
sums () {
  local -i n="$1" sum=0

  while ((n)); do
    d=n%10
    sum+=d*d
    n=n/10
  done

  echo "$sum"
}

is_happy () {
  local -i n="$1" seen=()

  while ((n != 1)); do
    if [ -n "${seen[$n]}" ]; then
      return 1
    fi
    seen[n]=1
    let n="$(sums "$n")"
  done

  return 0
}

happyNum () {
  if is_happy "$nums"; then
    echo "$nums" is a Happy Prime Number
  else
    echo Not Happy Prime Number
```

```
  fi
}

happyNum
```

Task4:

```
read -r -p "Enter three numbers: " -a arr

function one_g_two {
    if [ $((arr[0])) -gt $((arr[1])) ];then
        echo $((arr[0] - arr[1]))
    fi
}

function two_l_one {
    if [ $((arr[2])) -lt $((arr[1])) ];then
        echo $((arr[2] + arr[1]))
    fi
}

function one_eq_two {
    if [ $((arr[1])) -eq $((arr[2])) ];then
        echo $((arr[1] * arr[2]))
    fi
}

if [ ${#arr[*]} -gt 3 ]; then
    echo you have entered more than 10 characters
else
    one_g_two
    two_l_one
```

```
    one_eq_two
fi

#output should be like this
#3 4 1
```

Task5:

```
#!/usr/bin/env bash

read -r -p "Enter your numbers: " -a arr

echo Ascending Order
sorted=($(printf '%s\n' "${arr[@]}" | sort -n))
echo ${sorted[@]}

echo Descending Order
sorted=($(printf '%s\n' "${arr[@]}" | sort -n -r))
echo ${sorted[@]}

#output should be like this
# 1 4 5 6 8 1 10
```