CS 341 - Lab 03 - Using the Stack for Function Calls and Recursion

[5 Marks] Question 1. Convert the following C++ code to MIPS code. The function evaluate takes two arguments:
I. Pair x, y
II. Integers a,b,c which are the coefficients of polynomial $ax^2+bxy+cy^2$.
The functions evaluates the polynomial at x,y and returns the value.

```
struct mypair
{
      int x,y;
};

struct coeff
{
      int a,b, c;
};

int evaluate(struct mypair p, struct coeff cf)
{
      int a=cf.a, b=cf.b, c=cf.c;
      int x=p.x, y=p.y;
      return a*x*x+b*x*y+c*y*y;
}

int main()
{
      struct mypair p1; struct coeff c1;
      cout<<"Enter the pair x,y: ";
      cin>>p1.x>>p1.y;
      cout<<"Enter the coefficients a,b,c :";
      cin>>c1.a>>c1.b>>c1.c;
      cout<<"The value of polynomial at given point is:
"<<evaluate(p1,c1)<<endl;
}
```

Save the file as **evaluate.s**

This is how the program should behave:

```
Enter the pair x,y: 2
3
Enter the coefficients a,b,c:1
2
```

```
3
The value of polynomial at given point is: 43
```

Hint: The arguments of evaluate cannot all be accommodated in the 4 argument registers. This is intentional, so that you learn to use the stack for passing the additional argument.

Note: If you do not follow the standard MIPS conventions of caller and callee saved registers, and argument/return registers, you will get a maximum of 1 marks in this part.

===

[5 Marks] Question 2. Write a program which takes as input integer ‘n’ and outputs F(n) where:

```
F(n) =  n-10,          if n>100
      F(F(n+11)),      if n<=100
```

Do not try to simplify the function. Evaluate it using call to recursive function.
Save the file as **recursive.s**
This is how the program should behave:

```
Enter n: 99
The value of F(n) is: 91
```

Hint: You will find it useful to first write the C code version and then translate it(blindly). Use stack to save the register states before calling further functions.
Note: If you do not follow the standard MIPS conventions of caller and callee saved registers, and argument/return registers, you will get a maximum of 1 marks in this part.

==

[10 Marks] Question 3.
Consider the classes:

```
class vec
{
  public:
      int a,b;
};


class res
{
  public:
      int mag;
      int freq;
};
```

Write functions magnitude, maximumMagnitude with the following prototype:

```
int magnitude(vec v) {
      // return v.a*v.a + v.b*v.b
}


res maximumMagnitude(vec* A, int start, int end)
{
      //Returns the highest magnitude and its frequency in A[start,
end]
}
```

The function maximumMagnitude should call function magnitude internally. Subsequently, write the main function, which is also to be converted to assembly code, which will take as STDIN input the length and the elements for the array A of type vec, start index and end index(use 0 indexing) and call maximumMagnitude(A, start, end). It should then print the return values on STDOUT.
You may declare A globally and assume that it contains no more than 20 elements.
You may also assume that user enters correct input, so input checks are not necessary.
This is how the program should behave:

```
Input the length of array A: 5
Enter the elements of array:1
2
1
2
1
1
0
2
2
1
Enter start: 0
Enter end: 4
Maximum magnitude is: 5 and its frequency is:3
```

Here the program first takes the length of A (=5), then the elements ((1,2),(1,2),(1,1),(0,2),(2,1)) and start(=0), end(=4). It outputs the highest magnitude(5; mag(1,2)=mag(2,1)=5, mag(1,1)=2, mag(0,2)=4) and its frequency((1,2) appears twice,(2,1) appears once from 0 to 4)

Save the file as **magnitude.s**

Hint-1: First write a C-program, debug the logic, and then translate the same into assembly language.
Note: If you do not follow the standard MIPS conventions of caller and callee saved registers, and argument/return registers, you will get a maximum of 2 marks in this part.

Submission Instructions:

Submit a gzipped tar ball with the name <rollnumber>.tar.gz that has the 3 files evaluate.s, recursive.s and magnitude.s on Moodle. The assembly programs MUST be documented fully, else we reserve the right to not award any points.