Name: Pradyumn R Pai
Roll No: 50
Class: CS7A

PROGRAM CODE

ast.lex:

```
%{
#include "y.tab.h"
#include <string.h>
%}
%option noyywrap
%%
[0-9]+ { yylval.intval = atoi(yytext); return INTEGER; }
[a-zA-Z_][a-zA-Z0-9_]* {yylval.strval = strdup(yytext); return IDENTIFIER;}
[+\-*/();=] { return yytext[0]; }
. { /*Skip remaining characters */ }
%%
```

ast.y:

```
%{
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
%}

%type <nodeType> program
%type <nodeType> stmt
%type <nodeType> exp
%type <nodeType> term
%type <nodeType> factor
%token <strval> IDENTIFIER
%token <intval> INTEGER

%{
```

```
enum ASTKind {
   AST_INT,
   AST_VAR,
   AST_BINOP,
   AST_ASSIGN
};
struct AST {
   union ASTType{
      int intval;
      char *idname;
      char binop;
   } val;
   enum ASTKind kind;
   struct AST* left;
   struct AST* right;
};
%}

%union {
   int intval;
   char *strval;
   struct AST* nodeType;
}

%{
void addASTList(struct AST* n);
struct AST *make_int(int val);
struct AST *make_id(const char *s);
struct AST *make_binop(char op, struct AST *l, struct AST *r);
struct AST *make_assign(const char *name, struct AST *r);
%}
```

```
%%
program: /* empty*/ { $$ = NULL; }
   | program stmt; { $$ = NULL; print_ast($2,0);free_ast($2);}

stmt: exp ";" {$$ = $1;}
   | IDENTIFIER "=" exp ";" { $$ = make_assign($1, $3); };

exp: exp "+" term {$$ = make_binop('+',$1,$3);};
   | exp "-" term {$$ = make_binop('-',$1,$3);};
   | term {$$ = $1;};

term: term "*" factor {$$ = make_binop('*',$1,$3);};
   | term "/" factor {$$ = make_binop('/',$1,$3);};
   | factor {$$ = $1;};

factor: "(" exp ")" {$$ = $2;}
   | INTEGER {$$ = make_int($1);}
   | IDENTIFIER { $$ = make_id($1); };
%%
struct AST *make_int(int val){
   struct AST* n = malloc(sizeof(struct AST));
   n->left = NULL;
   n->right = NULL;
   n->val.intval = val;
   n->kind = AST_INT;
   return n;
}
struct AST *make_id(const char *s){
   struct AST* n = malloc(sizeof(struct AST));
   n->left = NULL;
```

```
    n->right = NULL;
    n->val.idname = strdup(s);
    n->kind = AST_VAR;
    return n;
}
struct AST *make_binop(char op, struct AST *l, struct AST *r){
    struct AST* n = malloc(sizeof(struct AST));
    n->left = l;
    n->right = r;
    n->val.binop = op;
    n->kind = AST_BINOP;
    return n;
}
struct AST *make_assign(const char *name, struct AST *r){
    struct AST* n = malloc(sizeof(struct AST));
    n->left = NULL;
    n->right = r;
    n->val.idname = name;
    n->kind = AST_ASSIGN;
    return n;
}

void print_ast(struct AST *a, int indent){
    if (!a) return;
    for (int i=0;i<indent;++i){
        printf("-");
    }
    switch(a->kind){
        case AST_BINOP:
            printf("Binop(%c)\n",a->val.binop);
            break;
```

```
        case AST_INT:
            printf("Int(%d)\n",a->val.intval);
            break;
        case AST_VAR:
            printf("Id(%s)\n",a->val.idname);
            break;
        case AST_ASSIGN:
            printf("Assign(%s)\n",a->val.idname);
    }
    print_ast(a->left,indent+2);
    print_ast(a->right,indent+2);
}

void free_ast(struct AST* a){
    if (!a) return;
    free_ast(a->left);
    free_ast(a->right);
    free(a);
}

void yyerror(char *s){
    printf("Error: %s\n",s);
}
int main() {
    yyparse();
    return 0;
}
int yywrap() {
    return 1;
}
```

OUTPUT:

x=5;y=10;z=x+y*3/5;

Assign(x)

--Int(5)

Assign(y)

--Int(10)

Assign(z)

--Binop(+)

----Id(x)

----Binop(/)

------Binop(*)

--------Id(y)

--------Int(3)

------Int(5)

a = 1+(2*3+(6/2)-1)*9;

Assign(a)

--Binop(+)

----Int(1)

----Binop(*)

------Binop(-)

--------Binop(+)

----------Binop(*)

----------Int(2)

----------Int(3)

----------Binop(/)

----------Int(6)

----------Int(2)

--------Int(1)

------Int(9)