1 文字列を答えなさい

バイト列が表す文字列を答えなさい．文字列に含まれる文字に割り振られたコードは次の表の通りである．なお，文字列に改行が含まれる場合は ↓ のように表記しなさい．(5 点× 4 問=20 点)

文字	ASCII	JIS 0201	JIS 0208	Unicode
A	0x41	0x41	-	U+0041
\	0x5c	-	-	U+005c
¥	-	0x5c	-	U+00a5
徳	-	-	0x4641	U+5fb3
山	-	-	0x3b33	U+5c71
↓	0x0a	0x0a	-	U+000a

1. ISO-2022-JP に符号化したとき以下のバイト列になる文字列

```
41 5c 1b 24 42 46 41 3b 33
1b 28 4a 5c 1b 28 42 0a
```

A\徳山¥↓

2. EUC-JP に符号化したとき以下のバイト列になる文字列

```
5c bb b3 41 c6 c1 0a
```

\山 A 徳↓

3. UTF-32BE に符号化したとき以下のバイト列になる文字列

```
00 00 5f b3 00 00 00 a5
00 00 00 41 00 00 5c 71
00 00 00 5c 00 00 00 0a
```

徳 ¥ A 山\↓

4. UTF-8 に符号化したとき以下のバイト列になる文字列

```
e5 b1 b1 5c 0a c2 a5 e5 be b3 41 0a
```

山\↓¥徳 A↓

2 実行結果を答えなさい

1. プログラム（p1）の実行結果（5 行）を答えなさい．(3 点× 5 行=15 点)

1: NULL
2: 1
3: 1
4: 3
5: 4

2. プログラム（p2）が正常に実行された時の実行結果を答えなさい．(6 点)

aaa

3. プログラム（p3）が正常に実行された時の実行結果を答えなさい．(6 点)

子
終了
親
終了

4. プログラム（p4）が正常に実行された時の実行結果を答えなさい．(6 点)

子
親
終了

5. プログラム（p5）が正常に実行された時の実行結果を答えなさい．(6 点)

x=0
x=1
x=2

6. プログラム（p6）が正常に実行された時の実行結果を答えなさい．(6 点)

x=1
x=1
x=1

3　プログラムを書換えなさい

1. プログラム (p7) が，execl() の代わりに execve() を使用するように，全体を書換えなさい．(15 点)

```
#include <stdio.h>
#include <unistd.h>

extern char **environ;
char *args[] = {"echo","aaa",NULL};
char *path="/bin/echo";

int main() {
  execve(path, args, environ);
  perror(path);
  return 1;
}
```

2. プログラム (myshell) は，授業で紹介したシェルプログラムです．次の実行例のように，コマンド行の先頭に代入形式の入力があった場合，環境変数を一時的に変更した上でコマンドを実行する機能を追加します．代入形式，コマンドの引数等の個数に制限は無いものとします．execute() 関数を完成しなさい．execvp() 関数に NULL を渡さないように注意してプログラミングすること．(10 点)

実行例

```
$ myshell
Command: printenv LC_TIME
C
Command: date
Thu Jul 27 10:29:03 JST 2017
Command: LC_TIME=ja_JP.UTF-8 date
2017 年 7 月 27 日 木曜日 10 時 29 分 12 秒 JST
Command: printenv LC_TIME
C
Command: LC_TIME=ja_JP.UTF-8 TZ=Cuba date
2017 年 7 月 26 日 水曜日 21 時 30 分 53 秒 CDT
Command:
$
```

```
void execute(char *args[]) {
  if (strcmp(args[0], "cd")==0) {
    ... 変更なし ...
  } else {
    int pid, status;
    if ((pid = fork()) < 0) {
      perror("fork");
      exit(1);
    }
    if (pid==0) {
```

```
      int i;
      for (i=0; args[i]!=NULL; i++) {
        if (putenv(args[i])<0) break;
      }
      if (args[i]!=NULL) {
        execvp(args[i], &args[i]);
        perror(args[i]);
      }
      exit(1);
```

```
    }
    while (wait(&status) != pid)
      ;
  }
}
```

3. プログラム (p8) はコマンド行で指定された 1 つのファイルに/bin/echo の実行結果を書込みます（実行例 1）．main() 関数だけ改造して，コマンド行で指定された複数のファイルに次々と/bin/echo の実行結果を書込むようにします（実行例 2）．改造した main() 関数を書きなさい．なお，エラー処理は省略してもよいものとします．(10 点)

```
1  実行例 1
2
3  $ p8 a.txt
4  $ cat a.txt
5  aaa
6  $
7
8  実行例 2
9  $ p8 a.txt b.txt     <--- ファイル数は制限なし
10 $ cat a.txt
11 aaa
12 $ cat b.txt
13 aaa
```

```
int main(int argc, char *argv[]) {
  for (int i=1; i<argc; i++) {
    int pid = fork();
    if (pid==0) {
      execEcho(argv[i]);
    } else {
      int stat;
      while (wait(&stat)!=pid)
        ;
    }
  }
  return 0;
}
```

4 プログラムリスト

リスト 1: p1 のソースプログラム

```
#include <stdio.h>
#include <stdlib.h>
void printEnv(char *val) {
  if (val!=NULL)
    printf("%s\n", val);
  else
    printf("NULL\n");
}
int main() {
  unsetenv("X");
  printEnv(getenv("X"));
  setenv("X", "1", 0);    // 0 に注意
  printEnv(getenv("X"));
  setenv("X", "2", 0);    // 0 に注意
  printEnv(getenv("X"));
  setenv("X", "3", 1);    // 1 に注意
  printEnv(getenv("X"));
  putenv("X=4");
  printEnv(getenv("X"));
  return 0;
}
```

リスト 2: p2 のソースプログラム

```
#include <stdio.h>
#include <unistd.h>
int main() {
  for (int i=0; i<2; i++) {
    execlp("echo","echo","aaa",NULL);
  }
  return 0;
}
```

リスト 3: p3 のソースプログラム

```
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>
int main() {
  int pid = fork();
  if (pid!=0) {
    int stat;
    wait(&stat);
    printf("親\n");
  } else {
    printf("子\n");
  }
  printf("終了\n");
  return 0;
}
```

リスト 4: p4 のソースプログラム

```
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>
int main() {
  int pid = fork();
  if (pid!=0) {
    int stat;
    wait(&stat);
    printf("親\n");
  } else {
    execlp("echo","echo","子",NULL);
  }
  printf("終了\n");
  return 0;
}
```

リスト 5: p5 のソースプログラム

```
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>
int main() {
  int stat, x = 0;
  for (int i=0; i<3; i++) {
    int pid = fork();
    if (pid==0) {
      printf("x=%d\n", x);
      return 0;
    } else {
      x++;
      wait(&stat);
    }
  }
  return 0;
}
```

リスト 6: p6 のソースプログラム

```
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>
int main() {
  int stat, x = 0;
  for (int i=0; i<3; i++) {
    int pid = fork();
    if (pid==0) {
      x++;
      printf("x=%d\n", x);
      return 0;
    } else {
      wait(&stat);
    }
  }
  return 0;
}
```

リスト 7: p7 のソースプログラム

```
#include <stdio.h>
#include <unistd.h>
char *path="/bin/echo";
int main() {
  execl(path, "echo", "aaa", NULL);
  perror(path);
  return 1;
}
```

リスト 8: p8 のソースプログラム

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
char *execpath="/bin/echo";
void execEcho(char *path) {
  close(1);
  int fd = open(path,
                O_WRONLY|O_CREAT|O_TRUNC,
                0644);
  if (fd<0) {
    perror(path);
    exit(1);
  }
  if (fd!=1) {
    fprintf(stderr, "何か変！\n");
    exit(1);
  }
  execl(execpath, "echo", "aaa", NULL);
  perror(execpath);
}
int main(int argc, char *argv[]) {
  if (argc>=2) {
    execEcho(argv[1]);
  }
  return 1;
}
```

リスト 9: myshell のソースプログラム

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/wait.h>
#include <ctype.h>
#define MAXLINE 1000
#define MAXARGS 60
int parse(char *p, char *args[]) {
  int i=0;
  for (;;) {
    while (isspace(*p))
      *p++ = '\0';
    if (*p=='\0' || i>=MAXARGS) break;
    args[i++] = p;
    while (*p!='\0' && !isspace(*p))
      p++;
  }
  args[i] = NULL;
  return *p=='\0';
}
void execute(char *args[]) {
  if (strcmp(args[0], "cd")==0) {
    if (args[1]==NULL)
      fprintf(stderr,"cd の引数が不足\n");
    else if (chdir(args[1])<0)
      perror(args[1]);
  } else {
    int pid, status;
    if ((pid = fork()) < 0) {
      perror("fork");
      exit(1);
    }
    if (pid==0) {
      execvp(args[0], args);
      perror(args[0]);
      exit(1);
    }
    while (wait(&status) != pid)
      ;
  }
}
int main() {
  char buf[MAXLINE+2];
  char *args[MAXARGS+1];
  for (;;) {
    printf("Command:␣");
    if (fgets(buf,MAXLINE+2,stdin)==NULL) {
      printf("\n");
      break;
    }
    if (index(buf, '\n')==NULL) {
      fprintf(stderr, "行が長すぎる\n");
      return 1;
    }
    if (!parse(buf,args)) {
      fprintf(stderr, "引数が多すぎる\n");
      continue;
    }
    if (args[0]!=NULL) execute(args);
  }
  return 0;
}
```